

# XML JOURNAL

THE ULTIMATE XML ENTERPRISE RESOURCE

April 2003 Volume: 4 Issue: 4

xml-journal.com

pg. 36



**XML Edge 2003 East**  
International XML Conference & Expo  
**SPECIAL SHOW REPORT**

## Guest Editorial

**XForms: Simplifying the Development of Transactional Web Apps**  
by David Litwack pg. 3

## Industry Commentary

**No Man Is an Island in the World of Pervasive Computing**  
by Paul Lipton pg. 5

## Reader Feedback

**XSLT for Code Generation in J2EE**  
pg. 7

## First Look

**XMLBeans**  
New from BEA  
by Hitesh Seth pg. 34

**XML News**  
pg. 48

**Industry Commentary**  
**Moving Toward Convergence**  
by John Magee pg. 50

\$6.99US \$7.99CAN



**SYS-CON MEDIA**



Part 3

**Creation and execution of the e-business dialogue**

Suhayl Masud 12

## Technology Trends: XML in Enterprise Apps

Hitesh Seth

Enterprise application vendors seek the benefits of investment in XML and Web services 7

## XSLT: What's New in XSLT 2.0

Dramatic expansion brings a new face to XML transformations

Jeff Kenton

18

## XSLT: XML-Coursebuilder, v.1.0

XML and XSLT format course modules into a comprehensive presentation

Mark Miller

22

## Feature: Struts and XSLT- It's Not an Either/Or Decision

Leverage the strengths of both to create dynamic configurable Web applications

Frank Neugebauer

26

## Information Assets: Leveraging Value of Disparate Information Assets

XML to the rescue

Kevin Huck

32

## XHTML + Voice: Versatile Multimodal Solutions

Rafah Hosn,

X+V 1.1: A look at what's new and why it's important Gerald McCobb, T.V. Raman 38

## XSL-FO: Modularize Formatting Objects

Learn how to create flexible, reusable Formatting Objects stylesheets

Frank Neugebauer

42

## Standards: MathML

Enabling mathematics to be served, received and processed on the Web

Ayesha Malik

46

# Macromedia

[www.macromedia.com/go/cfmxad](http://www.macromedia.com/go/cfmxad)

## FOUNDING EDITOR

Ajit Sagar [ajit@sys-con.com](mailto:ajit@sys-con.com)

## EDITORIAL ADVISORY BOARD

Graham Glass [graham@themindelectric.com](mailto:graham@themindelectric.com)Coco Jaenicke [cjaenicke@attbi.com](mailto:cjaenicke@attbi.com)Sean McGrath [sean.mcgrath@propylon.com](mailto:sean.mcgrath@propylon.com)Simeon Simeonov [talktosim@polarisventures.com](mailto:talktosim@polarisventures.com)

## EDITORIAL

## Editor-in-Chief

Hitesh Seth [hitesh@sys-con.com](mailto:hitesh@sys-con.com)

## Editorial Director

Jeremy Geelan [jeremy@sys-con.com](mailto:jeremy@sys-con.com)

## Managing Editor

Jennifer Stille [jennifer@sys-con.com](mailto:jennifer@sys-con.com)

## Editor

Nancy Valentine [nancy@sys-con.com](mailto:nancy@sys-con.com)

## Associate Editors

John Evdemon [jevdemon@sys-con.com](mailto:jevdemon@sys-con.com)Jamie Matusow [jamie@sys-con.com](mailto:jamie@sys-con.com)Gail Schultz [gail@sys-con.com](mailto:gail@sys-con.com)Jean Cassidy [jean@sys-con.com](mailto:jean@sys-con.com)

## PRODUCTION

## Production Consultant

Jim Morgan [jim@sys-con.com](mailto:jim@sys-con.com)

## Art Director

Alex Botero [alex@sys-con.com](mailto:alex@sys-con.com)

## Associate Art Directors

Louis F. Cuffari [louis@sys-con.com](mailto:louis@sys-con.com)Richard Silverberg [richards@sys-con.com](mailto:richards@sys-con.com)

## Assistant Art Director

Tami Beatty [tami@sys-con.com](mailto:tami@sys-con.com)

## CONTRIBUTORS TO THIS ISSUE

Rafah Hosn, Kevin Huck, Jeff Kenton, Paul Lipton,

David Litwack, John Magee, Ayesha Malik,

Suhayl Masud, Gerald McCobb, Mark Miller,

Frank Neugebauer, T.V. Raman, Hitesh Seth

## 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 © 2003 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.



# Simplifying the Development of Transactional Web Apps

WRITTEN BY DAVID LITWACK



The hardest part of writing transactional Web applications is finding a way to produce dynamic pages. The main underlying component of these pages, HTML forms, was added to what was originally a static, document-based standard, to allow the simple exchange of data between the user and the Web site. The more complex the information and the more sophisticated the interaction, the harder it's been to create these pages.

Worse, HTML was originally intended for browsers, but today we consume information in an ever-growing variety of ways. Attempts have been made to simplify the process, but nothing has fully addressed these issues because the underlying technology is too limited.

Until now – XForms, the next generation of forms to be included in the XHTML standard, and now a W3C Candidate Recommendation, improves on HTML forms by cleanly separating data, logic, and presentation. This new standard will not only make development better structured, but will also pave the way for a new generation of development tools.

XForms arrives as an ever-increasing percentage of information is moving across the Internet as XML. As the use of Web services increases, more business systems are being exposed using this standard. With new, easy-to-use tools to define XML integration, transformation, and mapping, there will soon be a huge repository of information and transactions available as distributed XML. The primary benefits of XForms are abstract description of presentation, independent of device; data binding between presentation components and XML instance data; and a range of interaction and logic capabilities without procedural programming.

XForms provides an abstract metadata description of presentation components such as selection lists and edit boxes. At runtime, this metadata is processed by “renderers” – server- or client-side components that translate the abstract to a specific implementation. As a result, XForms may be flexibly rendered in browsers by generating XHTML (either from the server or via a built-in or plug-in renderer), in rich clients by Java or Windows renderers, in specialized document formats such as PDF, and eventually by device-specific, vendor-supplied renderers in a variety of handheld devices.


In a world in which the way we connect and display information is constantly evolving, this moves the burden of adapting presentation from the developer to the vendor. At the recent XForms Implementation Workshop, some 20 such renderers were identified (see Novell's XForms preview at [www.novell.com/xforms](http://www.novell.com/xforms)).

Web services allow developers to abstract out the process of navigating and accessing information contained in a variety of physical stores and based on diverse technologies. The focus has been on simplifying the creation of services via the integration layer. Once the service is created, the metaphor becomes one of an XML request/response, with the processing in between becoming a black box, transparent to the consumer.

But this still leaves the burden on the developer to construct and deconstruct the XML. XForms presentation components may be bound to XML instance data, moving this burden to the underlying renderer. XForms will allow for end-to-end Web services, making the consumption of services as simple and high-level as the creation.

Most dramatically, XForms will enable the next generation of development tools to be more appropriate and productive for the mainstream business developer. Each generation of application architecture evolves from lower-level, early-adopter technologies to higher-level approaches that raise the level of abstraction. For example, client/server development tools provided increased productivity to that generation by making the definition of presentation visual, graphically generating SQL, and creating the relationship between the two. We've come up with many technologies to solve the equivalent Web problem: Perl, PHP, ASP, servlets, JSP, XSLT, and so on. None has been able to solve the issue of the complexity of forms development.

Combining Web services and XForms will enable the first high-level tools for developing services-oriented applications. While they will occasionally require dropping into lower-level code, the predominant development process will be at a higher level. It will have the feel of “XML programming” and will enable more rapid delivery of next-generation Web applications.

Web services, easily composed from existing information sources, combined with high-productivity tools for the rapid development of applications that consume XML – enabled by XForms – will be the key to dramatically increased business agility. 

## AUTHOR BIO

David Litwack is a senior vice president and a member of Novell's World-wide Management Committee responsible for Novell's Web Application Development Products. He previously served as president and chief executive officer of SilverStream Software, a company Novell acquired in July 2002. Before joining SilverStream, David served as executive vice president of Sybase Inc., and as president of Powersoft Corporation, prior to its acquisition by Sybase in February 1995.

DLITWACK@NOVELL.COM

# Mindreef

[www.mindreef.com](http://www.mindreef.com)



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

#### 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-Mussa megan@sys-con.com

Associate Sales Managers

Carrie Gebert carrieg@sys-con.com

Kristin Kuhnle kristin@sys-con.com

Alisa Catalano alisa@sys-con.com

#### SYS-CON EVENTS

President

Grisha Davida grisha@sys-con.com

Conference Manager

Michael Lynch mike@sys-con.com

Regional Sales Managers, Exhibits

Michael Pesick michael@sys-con.com

Richard Anderson richarda@sys-con.com

#### CUSTOMER RELATIONS

Senior Customer Care/Circulation Specialist

Niki Panagopoulos niki@sys-con.com

#### JDJ STORE

Manager

Rachel McGouran rachel@sys-con.com

#### WEB SERVICES

VP, Information Systems

Robert Diamond robert@sys-con.com

Web Designers

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

Online Editor

Lin Goetz lin@sys-con.com

#### ACCOUNTING

Accounts Receivable

Kerri Von Achen kerri@sys-con.com

Financial Analyst

Joan LaRose joan@sys-con.com

Accounts Payable

Betty White betty@sys-con.com

#### SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

1 888 303-5282

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

Cover Price: \$6.99/issue

Domestic: \$69.99/yr (12 issues)

Canada/Mexico: \$89.99/yr

all other countries \$99.99/yr

(U.S. Banks or Money Orders)

Back issues: \$12 U.S. \$15 all others

# No Man Is an Island in the World of Pervasive Computing

WRITTEN BY PAUL LIPTON



**D**o you want to understand our industry? Forget the big-name industry pundits and think-tanks. Look to the great poets like Donne and Shakespeare. You can't go wrong. The great poets can provide a long-term, human perspective on how we think, dream, and scheme. That insight is useful even in the new world of Web services and pervasive computing.

A better understanding of our own human nature will help us overcome our fear. This is important because too much fear can be both paralyzing and fatal, especially in IT. But, a tiny bit of fear can be helpful if it inspires us to continue moving and improving. Fear is often what compels us to keep up with new technology, to use it, and to sometimes benefit from it.

Recently, I noticed yet another consortium in the area of wireless. There certainly have been plenty of these, but this one is worth a glance. It's called The WLAN Smart Card Consortium. It's got some pretty big players in it. The idea is to help people use smart cards to access wireless access points around the world with one mechanism. Well, anything that helps the wireless industry overcome the twin roadblocks of ubiquity and security is a good idea in my book! But, whatever standard this group proposes or supports will be just one of many. The old joke that the nice thing about standards is that there are so many of them certainly applies to wireless. Even the tumult of competing Web services standards seems tame compared to the multiple consortia and technologies in the wireless arena.

So, why do these companies keep knocking their collective heads against the wall? Because, the current economy notwithstanding, wireless devices are becoming incredibly common and many, even the smallest, are starting to have significant computational power. These companies, and they include nearly every major hardware and software vendor, smell the money. Fear is also an important factor. Miss the market and you may perish – but nobody has yet figured out what the next killer application or dominant platform will be.

All this confusion and uncertainty has engendered increasing diversity of wireless device capabilities such as screen size, color depth, form factor, navigational technique (pen, buttons, keyboard, etc.), operating system, bandwidth and processor capability, and

browser markup language support. In short, wireless devices are not converging; they're diverging in many ways – sprouting cameras, voice recorders, midi-sound, and anything else that might possibly resonate with the diverse users of these devices.

How will smart IT managers cope and position themselves to meet this onslaught of client platform heterogeneity? A long time ago, smart managers realized that they couldn't stop the influx of PCs into the enterprise. Similarly, smart managers are realizing that they can't stop the influx of wireless devices. IT will be no more able to mandate the use of a particular client platform (such as a Palm Tungsten W or a Smartphone) than they can mandate the cell-phones that people use now. People will use what they like and what they can afford.

However, management can prepare for this onslaught. Decision makers can learn to bend with the wind by concentrating on just three core issues: quality of service, security, and delivery of information. That information is increasingly likely to be streams of XML data delivered by complex Web services, themselves orchestrated by other Web services. This XML data will be repeatedly transformed on multiple servers as part of the integration and orchestration required to fulfill the necessary business processes. And, ultimately, that XML data will be transformed one final time in order to deliver the appropriate XML vocabularies either to smart clients as SOAP messages or to thin clients using a markup language such as WAP2 or XHTML.

To transform that final XML data to a vocabulary appropriate for a particular wireless client will require the kind of personalization, aggregation, and client-sensing technology often associated with Enterprise Information Portals, the best of which are equipped with enhanced versions of XML transformational frameworks. Unfortunately, there are no dominant standards or industry leaders for such frameworks. However, several noteworthy frameworks exist that heavily leverage existing XML standards and widely used technologies in a relatively open fashion. The Apache Cocoon and AxiKit subprojects immediately come to mind, and solutions based on these and similar efforts may ultimately be safer, less-intimidating choices than more proprietary solutions.

Security and quality of service go right to the heart of our fears, and cannot be addressed solely at the Web services layer. Take the issue

~continued on page 21~

HOME



Enterprise Solutions



Content Management



Data Management



XML Labs



# PriceWaterHouse- Coopers

[www.pwcglobal.com/tech-forecast\\_syscon](http://www.pwcglobal.com/tech-forecast_syscon)

## XSLT for Code Generation in J2EE

I have applied XSLT for code generation in our J2EE-based NMS system. Currently 66 percent of code (\*.java) is generated. In addition, I generate SQL Schema, XML deployment descriptor, PDF documentation of the core object-relation model, message catalog for localization, and many more from XML-based models. Basically, our J2EE-based Network Management System (NMS) heavily uses "Model-Driven Programming" (MDP) with XSLT as a "Code Generation Scripting Language." If you're interested in learning more, check out "Model-Driven Programming," published in (*XML-J* Vol. 3, issue 8). The article proposed an extension of MVC architecture and showed how to implement in software development projects.

sarkar\_soumen

## Wanted: Web Services Definition

To define Web services in terms of repackaged RPC is to imply a synchronous coupling which is not necessarily so, in fact the Web service model is essentially uncoupled. In SOA terms one can regard Web services as being Web-facing objects or pseudo-objects which can discover and interact with each other by commonly accepted protocols, the most central of which is XML.

Gervas Dougl as

There is a lot of information on XML Web services, and everybody will have their own way of defining Web services. Here's mine:

XML Web services facilitate seamless and secure communication of two applications over Internet, without any platform issues. For instance, a Perl application running on a Linux machine can send a (XML) request to a .NET application running on a Windows 2000 machine over some Web protocol (such as HTTP), and get back the (XML) response.

XML, WSDL, UDDI, etc., all are crucial to Web services, as well as SAML, and the GXA suite of standards.

I think it would be interesting to find out how many people are using the SOAP approach to XML Web services and how many are going with the REST approach.

Darshan Singh

Letters may be edited for grammar and clarity as well as length. Please e-mail any comments to Hitesh Seth (hitesh@sys-con.com).

# XML in Enterprise Applications

WRITTEN BY HITESH SETH



Enterprise application systems – ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), SCM (Supply Chain Management), etc. – have for decades run mission-critical systems for medium to large organizations. With their large breadth of integrated, prebuilt functionality these systems have been at the heart of the IT infrastructure. There's also been the constant challenge of integrating them with legacy/custom/homegrown systems as well as the e-business/edge applications and infrastructure. As architects and developers, we've spent sleepless nights trying to understand application-specific APIs, interface tables, file formats, and so on. The era of XML and Web services has given us hope, and we're seeing a movement toward decoupled interfaces based on XML messaging and eventually Web services.

To appreciate the value that XML brings to enterprise applications, I spoke with technology executives of leading enterprise application vendors who have started to see the benefits of their investments in technologies such as XML and Web services.

## From Flat Files and APIs to XML and Web Services

Most application providers started providing interfaces to their applications using file exchange. Files in a particular format (which was often proprietary to the application vendor) were expected to be created by external applications and were to be consumed by the application; similarly, on the outbound side they were created for consumption. With the advent of EDI, standard file formats based on EDI (with a lot of variations, of course) began to be utilized. To facilitate online or real-time integration, vendors then began exposing key abstractions of their functionality as Remote Procedure Calls (RPC). These were typically developed in C/C++ and ported across the various platforms supported by the vendor. In addition to these tightly coupled interfaces, a number of vendors also provided database drivers that typically allowed "read-only" access to the back-end data (as business rules to validate the data were typically encapsulated in either the GUI or the business functions). Another mechanism was that of an interface table, which allowed external applications to deposit external data into a set of RDBMS tables; it was then processed by "processors" specific to that business transaction.

In late '90s and early 2000s XML came into existence. Application vendors saw this as an opportunity to provide technology-independent interfaces to their systems. Through either internal product development or OEM-bundling arrangements of "XML Integration Servers," they delivered XML interfaces to their existing application suites. This is where we are today, with most applications in production. Most application providers, however, have further embraced the service-oriented architecture (SOA) concepts and have architected a Web services strategy to provide industry-standard Web services interfaces to their back-end functionality.

## .NET and/or Java

At some point most application providers had to decide whether to inherently utilize a Java- or .NET-based architecture. Web services gives application providers an easy solution for interoperability with customers' application development, enterprise portals, integration, etc., infrastructures. Web services and XML decouple them from their core technologies so that even if they're built on top of .NET or Java, they can support both environments and many more.

## Interoperability Within Multiple Versions

Over the past two decades the functionality, architecture, technologies, and even the paradigm utilized by application vendors have evolved. For example, there are many versions of SAP applications in production today. Some are based on mainframe technologies, some are client/server, and some utilize the next-generation NetWeaver platform. In many cases these applications have to talk with each other or with another SAP "bolt-on" application on the base ERP, such as CRM or SCM. In these scenarios Web services and XML provide a mechanism through which a wrapper/interface can be developed that uses a predefined XML-based message or transaction and consistently works with multiple versions of the underlying application through version-specific bindings.

## Not Just Internal Services

The scope of Web services is definitely not limited to inbound connectivity to enterprise applications. In a number of scenarios (for example a credit card payment or credit checks through an external agency), an enterprise application needs to connect with the external

application from within a business process. Web services can provide a seamless way of connecting with external providers in a standard fashion. Most application vendors support Web services today, but typically through COM/Java wrappers. For most packages, out-of-box support for outbound Web services into the business processes of these applications remains to be seen.

### UDDI as an Application Business Registry

With the emergence of Web services a public registry based on UDDI was touted as a key highlight. For a number of reasons, UDDI-based public registries aren't as successful today, but the

concept of having a business services registry is intriguing. Providers of ERP/CRM applications offer a lot of functionality, and with most of their components now exposed as Web services, UDDI or a UDDI-like framework allows creation of an application business registry in which metadata and information for all the services provided by the application can be stored and exposed to the rest of the enterprise for use and consumption by other Web services consumers.

### Composite Applications

Web services provide a lot of technology-related benefits. From a business perspective the biggest advantage that Web services-based architectures

bring is business agility. Applications designed and developed using the services-oriented model are more capable of accepting change than traditional tightly coupled architectures. In the world of enterprise applications (and in enterprise integration), a new term is emerging: "composite applications." Composite applications are a new breed of applications that sit on top of existing applications. They contain new business processes and functionality. Instead of being stand-alone applications, however, they also utilize functionality in existing systems. An example of a composite application is a merger application. An SOA makes the development of these composite applications (or bolt-ons) much easier.

## XML AND WEB SERVICES SUPPORT

- |                     |   |
|---------------------|---|
| <b>J.D. Edwards</b> | <ul style="list-style-type: none"> <li>Connectivity with J.D. Edwards has evolved from interface tables (popularly known as Z-tables) and Java APIs to XML.</li> <li>The company provides XML/Web services-based integration with the various business processes/services using XPI Technology.</li> <li>A business process modeling tool is evolving to support BPEL.</li> </ul>   |
| <b>Lawson</b>       | <ul style="list-style-type: none"> <li>Connectivity with Lawson has evolved to support technologies such as COM+, OLE-DB data provider, Java, and the application gateway service, which is based on XML.</li> <li>Lawson has utilized their existing core metadata for their applications (4,000+ screen definitions) and used that in conjunction with XML/XSL to create dynamic user interfaces (portals).</li> </ul>  |
| <b>Microsoft</b>    | <ul style="list-style-type: none"> <li>Both Navision and Great Plains (part of Microsoft's Business Solutions group) utilize BizTalk Server to provide XML/Web services interfaces – Navision (Commerce Gateway), Great Plains (eConnect).</li> <li>Currently, an architecture effort is underway to create the underlying architecture and components for the next-generation application suite, which will be based on Microsoft .NET, including .NET Servers, Windows/SQL Server/BizTalk/Exchange, and integrated with Microsoft Office.</li> <li>Microsoft CRM applications are already built up on top of the Microsoft .NET platform.</li> </ul>  |
| <b>Oracle</b>       | <ul style="list-style-type: none"> <li>Connectivity with Oracle Applications (now known as the e-Business Suite) has evolved from using database-based open interface tables to Oracle Advanced Queuing (AQ), which now has the capability to send/receive XML-based transactions.</li> <li>Oracle has extensively leveraged the Business Object Definitions (BODs) defined by Open Applications Group (as OAGIS).</li> </ul>   |
| <b>PeopleSoft</b>   | <ul style="list-style-type: none"> <li>Connectivity with PeopleSoft applications has evolved from a screen scraping-based message agent to Tuxedo/JOLT-based RPC mechanisms, to HTML/XML interfaces, and now a complete XML-based messaging framework (using PeopleSoft Integration Broker in PeopleTools version 8.4).</li> <li>PeopleSoft delivers hundreds of Enterprise Integration Points, which have defined XML messages and are accessed via Web services.</li> <li>Support is provided for both synchronous and asynchronous invocation mechanisms.</li> </ul>   |
| <b>SAP</b>          | <ul style="list-style-type: none"> <li>Connectivity with R/3 has evolved to support multiple interaction models – IDOCs, which are based on EDI-like document exchanges; RFC (Remote Function Call), which is a tightly coupled RPC-based mechanism; JCA/.NET connectors to XML; and now Web services.</li> <li>SOAP/HTTP are currently supported through the Internet Communication Manager component in the J2EE SAP Web Application Server (v 6.20).</li> <li>The company recently announced a large-scale enterprise service architecture initiative/platform called NetWeaver.</li> <li>XML is used extensively with the various components of the NetWeaver platform – delivering portals, information, and business process integration.</li> <li>Interoperability with J2EE/.NET Frameworks is a key highlight of NetWeaver.</li> </ul> |
| <b>Siebel</b>       | <ul style="list-style-type: none"> <li>Connectivity with Siebel has evolved from a tightly coupled COM Business Object Framework to XML and Web services.</li> <li>Siebel has utilized XML for representing user interfaces and portal integration since Siebel 7.</li> <li>Siebel 7.5 already supports SOAP/WSDL.</li> <li>Siebel Universal Application Network (UAN) provides end-to-end business processors, integrating multiple applications, and supports execution on top of multiple integration brokers.</li> <li>UAN extensively leverages XML/XML Schemas, WSDL, XSLT, and XML-based flow language, which is evolving toward BPEL.</li> </ul>  |



# PolarLake

[www.polarlake.com](http://www.polarlake.com)

**Mike Madden**  
CTO  
J.D. Edwards

*J.D. Edwards' middleware solutions, XPI (eXtended Process Integration) and XBPs (eXtended Business Processes), give customers the power to collaborate between applications and among enterprises, regardless of software packages utilized. XPI and XBPs allow customers to rapidly react to marketplace changes by providing an application integration process that incorporates an emerging XML and Web services strategy and, unlike competing offerings, is not enormously costly, complex, and time consuming. J.D. Edwards middleware solutions also help customers maximize the application investments they have made with J.D. Edwards and move quickly and effectively, but also safely and securely, down the path of ERP II.*



**Kyle Gunderson**  
Development Director  
of Web Technologies  
Lawson Software

*XML technology was instrumental in producing Lawson's browser-based, thin-client Portal solution. By using XML to extend our system-definition metadata to a rich set of Web-tier services, we dramatically shortened the development cycle while enhancing the functionality of our Internet-based Portal and Design Studio solutions.*



**Marcus Schmidt**  
Lead Product Manager  
Microsoft Business Solutions

*Microsoft Business Solutions is using XML Web services today in projects such as its forthcoming business network, which will enable companies to streamline collaboration with customers, suppliers, and other business partners. XML Web services are integral to every future product we build, and a foundation component for our next-generation applications.*



**Art Kruk**  
VP, Research & Technology  
for Oracle E-Business Suite  
Oracle Corporation

*Oracle has built and is continuing to build industry-standard messaging capabilities directly into our base applications and technology products. And, not only are we building to standards, we are helping to define those standards. Oracle has contributed significantly to the various standards-based content subcommittees (OAG, ebXML, RosettaNet).*

*Today, Oracle has approached the delivery of Web services from three perspectives. First, the ability to define and automatically create any business object as a Web service. Second, the ability to manage, transform/map, and deliver A2A and B2B integration messages, including Web services, XML, RosettaNet, and others. Third, direct message integration into the integration layer of our E-Business Suite applications.*

*Today, thousands of Oracle E-Business Suite business objects can be enabled as Web services using Oracle's JDeveloper toolset and Oracle9iAS integration services. In addition, to solve a class of specific customer pain points, Oracle has prebuilt 6 key business objects consisting of 53 Web services. Going forward, Oracle plans to offer more prebuilt Web services in areas where customers show specific interest.*

**David Sayed**  
Marketing Manager,  
PeopleSoft Technology  
PeopleSoft, Inc.

*Web services build on top of the standards-based integration framework that is an integral part of the PeopleSoft Internet Architecture, and all PeopleSoft applications.*

*The support and wide deployment of Web services are key factors in ensuring interoperability between enterprise applications. Emerging Web services standards will play an important role in the seamless deployment of business processes that span multiple applications. PeopleSoft's support of current and emerging Web services standards, in addition to supporting traditional integration approaches such as batch, enables us to address the real needs of our customers.*



**Thomas Mattern**  
Product Marketing Manager  
SAP

*SAP expects a significant move from three-tier C/S architecture and basic Internet technology to an Enterprise Services Architecture (ESA), which expands the concept of Web services into a services-based, enterprise-scale business architecture. While Web services are merely a technical concept, the ESA is the blueprint for services-based business solutions, encompassing internal and external SAP and non-SAP systems relevant to a business process in the value chain. These solutions are expected to set new standards in usability, scalability, and adaptability while significantly lowering total cost of ownership.*



**Doug Smith**  
VP, Architecture  
Siebel Systems

*XML standards are quickly leading to an inflection point in how enterprise applications are developed and deployed. Siebel Systems has committed to using XML standards throughout our offerings. Siebel 7.5 is currently in market and provides native support for XML, SOAP, and WSDL for exposing the user interface, business services, and customer data. Siebel's Universal Application Network, which provides integration applications, or out-of-the-box business processes, is based upon XML standards such as WSFL, XSD, and XSLT. Siebel's UAN and the underlying XML technologies enable customers to address end-to-end business processes that can be readily deployed and adapted to changing business conditions.*

## Conclusion

XML and Web services architectures provided an evolutionary path to the integration infrastructure for enterprise applications. Most of the large application vendors have utilized these technologies today or have them in their near-future technology plans. This is good for application developers and integrators because we can focus on the real business logic rather than rattle with a bunch of proprietary APIs. With the emergence of business process modeling and execution standards such as BP4WS and WSCI, we're bound to see the continued infiltration of Web services technologies into enterprise applications. ☛

HITESH@SYS-CON.COM

Premiering  
June 2003  
at  
**JavaOne**  
Oracle's 2003 Worldwide Java Developer Conference

www.sys-con.com



# Millions of Linux Users One Magazine

## Linux Business & Technology

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *Linux Business & Technology* is aimed squarely at providing this group with the knowledge and background that will allow them to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *Linux Business & Technology* will not feature low-level code snippets but will focus instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month will see a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

### Regular features will include:

*Advice on Linux Infrastructure*  
*Detailed Software Reviews*  
*Migration Advice*  
*Hardware Advice*  
*CEO Guest Editorials*  
*Recruiting/Certification Advice*  
*Latest News That Matters*  
*Case Studies*

**LINUXEDGE**  
conference & expo

June 3-5.....LONDON  
June 24-26.....BERLIN  
September.....HONG KONG  
October.....CALIFORNIA

**SYS-CON  
MEDIA**

The World's Leading Technology Publisher

FOR ADVERTISING INFORMATION:

CALL 201.802.3020 OR  
VISIT [WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)

# SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY

# \$49.99

12 ISSUES/YR

\*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

## SUBSCRIBE TODAY!

[WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)

OR CALL

1-888-303-5282

ALL BRAND AND PRODUCT NAMES USED ON THIS PAGE ARE TRADE NAMES, SERVICE MARKS, OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES



# Building a Real-World Web Service –

Part 3

WRITTEN BY  
SUHAYL MASUD

*Putting the future in perspective*

**W**eb services are the new “it” in the IT world, and vendors are rushing in to stake claims in this landscape, each with a different marketing spin on how they “do Web services.” However, simply sending SOAP-based messages between machines is not really “doing Web services”; this is a limited view that obscures the bigger picture.

This series of articles demonstrates what a real-world Web service is and how to build one using features of Web services with components and knowledge of RosettaNet, an industry leader in e-business process standards. Having defined the WSDL definitions and constructed an abstract process in BPWS4J in the first two installments, I now focus on implementation. This article is all about implementation and construction...bring your hard hat along.

## Using the Engine: BPWS4J

This installment demonstrates the creation and execution of a simple version of the e-business dialogues we’ve been building in this series. I’ll create a bunch of BPEL, WSDL, and Java files using the BPWS4J Editor, and Eclipse, and then execute them on the runtime engine, BPWS4J. I’ll also show you how to install all the tools and components necessary to create and run e-business dialogues for yourself. The complete source code for this article is available at [www.sys-con.com/xml/sourcececfm](http://www.sys-con.com/xml/sourcececfm).

## Setting up the Environment

Before I talk about the business dialogue we’re constructing in this article, I want to dive right into setting up the environment. I encourage you to take a hands-on approach and experiment with the environment as you work through the article. The purpose is to set up an environment to develop and execute asynchronous business processes to perform the e-business dialogue we’ve been developing. Here is a list of requirements for this task:

- **Java SDK from Sun Microsystems:** <http://java.sun.com/j2se>. I used J2SE 1.4.1.
- **Eclipse, an open-source IDE from Eclipse.org:** [www.eclipse.org/downloads/index.php](http://www.eclipse.org/downloads/index.php). I used version 2.0.2.
- **Apache Tomcat 4.1, an open-source server solution for servlets and JSPs:** <http://jakarta.apache.org/tomcat/index.html>
- **BPWS4J Engine and Editor from IBM alphaWorks:** [www.alphaworks.ibm.com/aw.nsf/download/bpws4j](http://www.alphaworks.ibm.com/aw.nsf/download/bpws4j)

- **JavaBeans Activation activation.jar:** <http://java.sun.com/products/javabeans/glasgow/jaf.html>
- **JavaMail mail.jar:** <http://java.sun.com/products/javamail>

To run Eclipse, Tomcat, or the BPWS4J engine, we need the Java runtime. We’ll use Eclipse to write, compile, and execute any Java classes that we build along the way. To create a process, we’ll use the BPEL editor, which is available as an Eclipse plug-in. To run a business process we need the BPWS4J engine, which runs on Tomcat. Finally, the activation and mail APIs are used for communication between the engine and clients.

## Installing the Components

The first step is to install the Java SDK; just click on the installer file you downloaded and follow instructions. Test the installation by typing `java -version` in a command/shell window. You should see:

```
java version "1.4.1"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1-b21)
Java HotSpot(TM) Client VM (build 1.4.1-b21, mixed mode)
```

The next step is to install the Java Activation Framework. I simply removed activation.jar from the jaf-1\_0\_2.zip file I’d downloaded and placed it in the `jre/lib/ext` directory. To install JavaMail, I opened up the javamail-1\_3.zip file `imap.jar`, `mail.jar`, `smtp.jar`, `pop3.jar`, and `mailapi.jar` into `/jre/lib/ext`. These files are automatically picked up by the Java runtime.

Next we install Apache Tomcat. The installation is straightforward; just follow the steps in the installation wizard. To test the installation, start up Tomcat and type `http://localhost:8080/` in your Web browser. If the installation is working, you should see the Apache administration page.

Next we install Eclipse, which is a breeze as well; the install program finds the Java runtime and handles the necessary properties. If Eclipse doesn’t find the runtime automatically, you might need to reboot to update the environment and install Eclipse again.

We’ll install the BPWS4J editor, which is an Eclipse plug-in and needs to be deployed into `/eclipse/plugin` directory. To do this, unzip the contents of the BPWS4J Editor zip file into your `/eclipse` folder – the contents will be extracted into `/eclipse/plugins`. That’s

it, you're done installing the editor. Now you can launch Eclipse and open up a "BPWS Perspective." Once you open a .bpel file, or create a new one, you should see BPWL constructs on the toolbar. The editor lets you create BPWL processes in a visual environment as well as a source code view.

It's now time to install the BPWL engine. This is a fairly simple task as well: extract the contents of the zip file to a directory of your choosing and simply grab the bpws4j.war file from this directory, then drop it into the Tomcat4.1/webapps directory. To test the installation, start the Tomcat server (see Figure 1), and point the browser to <http://localhost:8080/bpws4j>. If the installation worked, you should see a Web page that allows you to run the admin client. To ensure that the server is configured properly, point the browser to <http://localhost:8080/bpws4j/soapprcrouter>. You should see a message stating: "Sorry, I don't speak via HTTP GET - you have to use HTTP POST to talk to me". You should test the installation further and smooth out any problems by working with the samples provided with the BPWS4J package. Simply drop the bpws4j-samples.war file into the Tomcat4.1/webapps directory, and load the samples package in your Eclipse project. Follow the samples installation instructions for each process and you should soon have the environment warmed up for the development task ahead.

## Easing the Development Process

We're working with Java, WSDL, and BPWL code. Since BPWL, WSDL, and especially the BPWS4J engine are all young efforts, sophisticated integrated development environments are a few months away. For now you'll have to get your hands dirty working with the nuts and bolts of code, but I have three tips to make the development process easier.

First, make full use of the logger tool log4j, an indispensable tool developed by Apache and shipped with the BPWS4J engine. Your BPWS4J engine will log errors, information, and warnings into a log file using this tool, and this data is critical to understanding how your BPWL process is working. To enable the logger, go to where you installed BPWS4J on Tomcat, navigate to the bpws4j\WEB-INF\classes\log4j.properties file, and insert the lines shown below into the file. The engine will start logging into a file named bpel.log:

```
log4j.rootLogger=DEBUG, R
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=bpel.log
log4j.appender.R.MaxFileSize=100KB
log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%p %t %c - %m%n
```

Second, try to simulate partner processes on different machines. If that isn't possible and you have to use the same machine, at least run separate instances of the BPWS4J engine on different instances of Tomcat. This makes it easy to follow code execution on each engine.

Third, use Eclipse to do all your code development, with the BPWL editor integrated into the environment. It becomes easy to debug and run Java and BPWL files.

## The End-to-End Purchase Order Scenario

It's now time to explore the e-business dialogue we're constructing. Figure 2 shows an end-to-end scenario of how a purchase order process would work in a BPWL environment at both ACME and Laptops, Inc. In this scenario, ACME and Laptops, Inc., are communicating publicly. ACME places a purchase order request and first receives a receipt acknowledging that Laptops,

Inc., has received the purchase order request. Then ACME receives a purchase order acceptance from Laptops, Inc., which ACME acknowledges with a receipt message as well. On the private side, the processes at both ACME and Laptops, Inc., are interacting with internal Web services and Java applications.

Let's look at what the processes are doing on the private side at both ACME and Laptops, Inc. At ACME the purchase order process is triggered by an Inventory Manager Java application. The process completes execution by sending the results of the execution to an internal Web service. On the Laptops, Inc., side the process is started by the PO request received from ACME. The process invokes an internal Web service to get the acceptance for the purchase order and finally wraps up by sending the results of the process to another internal Web service. There are 12 components in this scenario, 2 BPWL processes, 4 internal Web services, 4 Java apps, and 2 public Web services. The scenario describes reliable asynchronous messaging between the partners using actual RosettaNet purchase order processes.

We'll build the public interactions between ACME and Laptops, Inc. (components 1-6 in Figure 2), leaving the private interactions to be completed in the next article. I've simplified the business scenario and will properly cover how the processes work as well as how WSDL, BPWL, and the runtime engine interact before going further and creating the rest of the components.

## Building the Public Web Service Definitions

I'll start by building the WSDL definitions for the one private and two public Web services, starting with the poRequester.wsdl Web service (component 2 in Figure 2) to be deployed on ACME. This Web service allows the Inventory Manager to replenish inventory by placing a purchase order. For more details on working with WSDL, refer to Part 1 of this series (*XML-J* Vol. 4, issue 2). BPWL-enabled WSDL definitions require some additional information and since this WSDL file will be used in the BPWL process, it requires a service link definition to be added to the file. Service links enable partners in the BPWL process to be linked to actual "actions" defined in the Web service. Another deviation from the normal WSDL definition is that we don't have to describe the binding section for the WSDL definition - BPWS4J automatically generates the bindings necessary to interact with the defined port types. The poRequester.wsdl file begins with the namespace declarations:

```
<definitions targetNamespace="http://www.acme.com/
services/poRequester"

xmlns:ACME="http://www.acme.com/services/poRequester"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:slt="http://schemas.xmlsoap.org/ws/2002
```

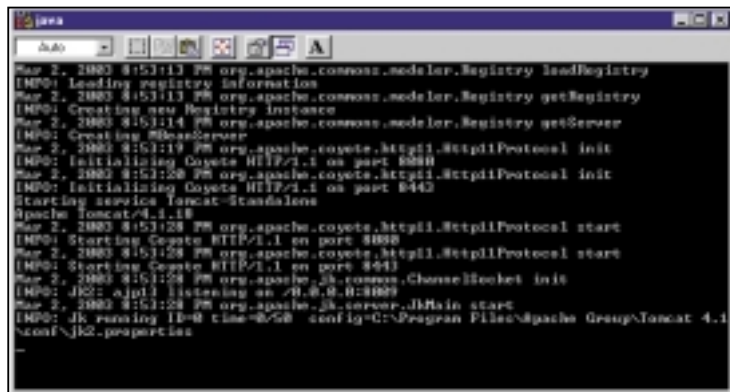


Figure 1 • Running Tomcat instance



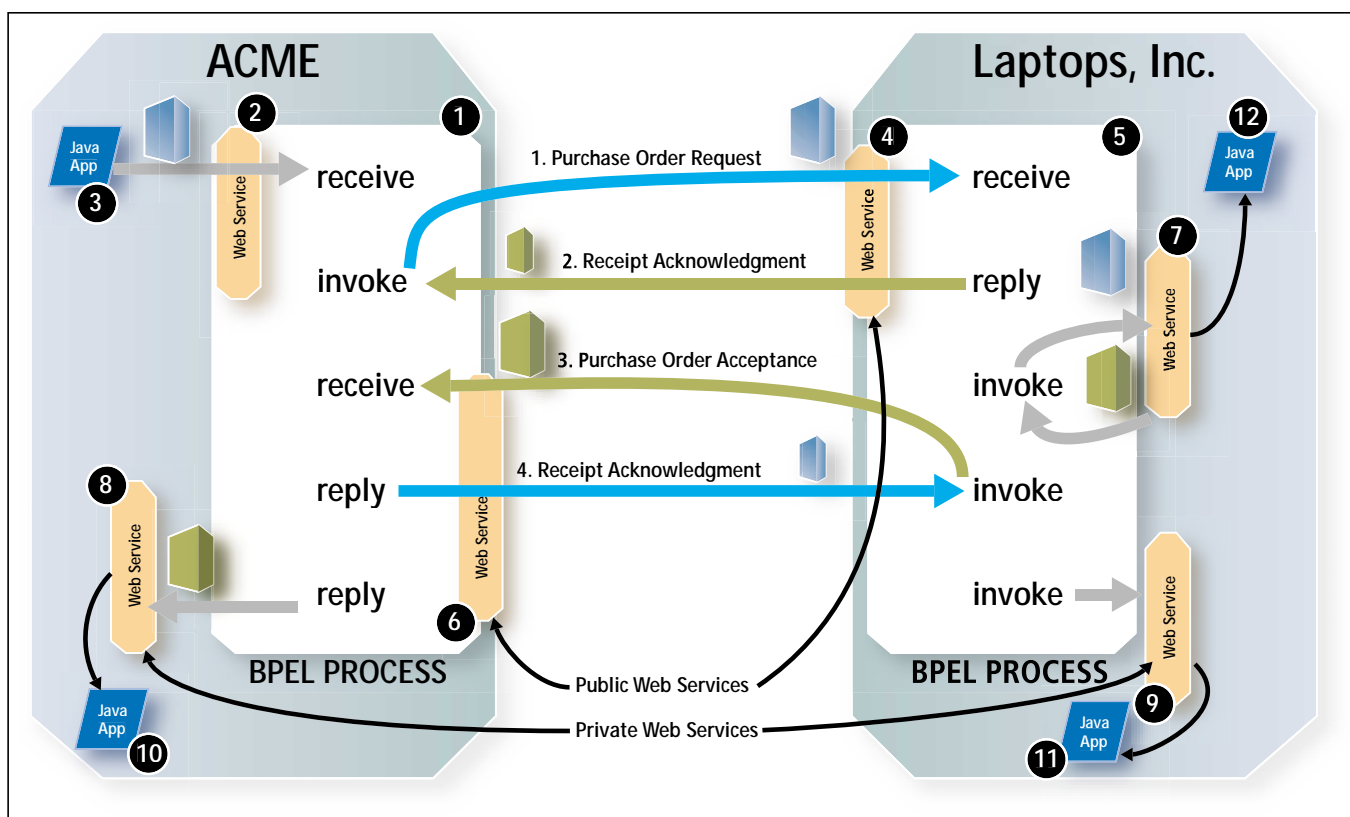


Figure 2 • An end-to-end e-business dialogue scenario using BPEL and WSDL components

```
/07/service-link/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
```

There are two messages in this Web service: a request for purchase order, and a message that acknowledges the receipt of the request. In the complete scenario, the replenish PO message is a complete RosettaNet purchase order, and the receipt message is also more elaborate and contains a copy of the message it is acknowledging. These features will be added in the next installment, but for now we'll keep both messages simple.

```
<message name="ReplenishRequestType">
  <part name="replenishPO" type="xsd:string"/>
</message>
<message name="ReceiptAckType">
  <part name="receiptAck" type="xsd:string"/>
</message>
```

Next, define the port types and the collection of operations that the port type supports. This is a very simple service that has only one operation. When the Inventory Manager interacts with the Web service, the Web service receives an incoming replenish request and sends a receipt acknowledgment back to the Inventory Manager.

```
<portType name="replenishRequestPort">
  <operation name="replenishRequest">
    <input message="ACME:ReplenishRequestType"/>
    <output message="ACME:ReceiptAckType"/>
  </operation>
</portType>
```

The BPEL process defines capabilities of partners by using service links that link a partner to a port type and a set of operations in the WSDL file. For more information on service links,

please refer to Part 2 of this series (*XML-J* Vol. 4, issue 3). Here is the service link definition:

```
<slt:serviceLinkType name="replenishRequestSLT">
  <slt:role name="inventoryService">
    <slt:portType name="ACME:replenishRequestPort"/>
  </slt:role>
</slt:serviceLinkType>
```

In a typical WSDL definition the next section is for defining the binding information that specifies the format of the messages sent to the Web service, and the address to send the messages to. However, since we're deploying this Web service as an associated component of a BPEL process, the BPWS4J engine generates the necessary bindings so the BPEL process can absorb the Web service, listening to the defined ports for any activity. Therefore, the bindings and service sections of the WSDL definition are empty.

```
<service name="ACMEPORequesterServiceBP">
</service>
</definitions>
```

The public Laptops, Inc., Web service, `LaptopsIncPlacePO.wsdl` (component 4 in Figure 2), takes a purchase order request as input and sends a receipt acknowledgment as a response. It has a structure very similar to the Web service we just defined (see Listing 1).

Let's reflect on the public process: ACME sends a purchase order request with Laptops, Inc., and receives a receipt message. Next, Laptops, Inc., needs to send ACME a substantive purchase order acceptance response. To receive this PO acceptance message, define the `poAcceptanceReceiver.wsdl` Web service on ACME (component 6 in Figure 2). Its purpose is to receive a PO acceptance and to send a response acknowledging receipt of the PO acceptance. The definition is pretty straightforward and follows the same logic as the previous two definitions (see Listing 2).

Now that we've defined the WSDL definitions, the next step is to create the BPEL processes that will use these Web service definitions to conduct business processes.

## Building the BPEL Processes

Conceptually, we need to build only two BPEL processes, one for ACME and another for Laptops, Inc. However, due to a minor bug in the early alpha version of the BPWS4J engine, I have to split the process for ACME into two separate BPEL processes. IBM will hopefully fix this minor nuisance soon.

The first BPEL process at ACME is ACMEPORequester.bpel (the first half of component 1 in Figure 2); it's layered on top of the poRequester.wsdl created earlier. This process takes a PO request from an Inventory Manager, and then sends that request to Laptops, Inc., by invoking the placePORequest operation on the Web service hosted by Laptops, Inc. The process passes the result of this invocation, just a receipt acknowledgment for our simple scenario, back to the Inventory Manager.

The second BPEL process on ACME is ACMEPOReceiver.bpel (second half of component 1 in Figure 2), layered on top of poAcceptanceReceiver.wsdl. This process receives the PO acceptance message sent by Laptops, Inc., and sends back a receipt message to Laptops, Inc.

Laptops, Inc., hosts the third BPEL process (component 5 in Figure 2); it receives a PO request (from ACMEPORequester process) and sends a receipt message to the requester, and then it sends the requester (ACMEPOReceiver process) a PO Acceptance message, and receives a receipt message.

During the construction and testing phase of the BPEL processes, it's important to examine the log files on both BPWS4J engines to gain insight into how the processes are working. I'll next define ACMEPORequester.bpel (see Figure 3). The process definition begins with a namespace declaration:

```
<process name="acmePORequesterProcess"
  targetNamespace="http://www.acme.com/services/acmePORequesterProcess"
  xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/"

  xmlns:acmePOR="http://www.acme.com/services/poRequester"
  xmlns:laptopsPO="http://localhost:8080/laptops/LaptopsIncPlacePO.wsdl">
```

Next I define the partners of the process. There are two partners: the first is the Inventory Manager, defined as the inventory service in the definition below. The Inventory Manager interacts with the process through operations defined in the "replenishRequestSLT" service link. The second partner plays the role of a PO fulfiller in this process, and the partner definition allows the process to interact with Laptops, Inc., through the "PORequestSLT" service link defined in the Laptops, Inc., WSDL file.

```
<partners>
  <partner name="inventoryService" serviceLinkType=
    "acmePOR:replenishRequestSLT"/>
  <partner name="POFulfiller" serviceLinkType=
    "laptopsPO:PORequestSLT"/>
</partners>
```

I'll build the containers required to contain the data the process is receiving and sending. There are four containers in this process – the first two are used to receive from and respond to the Inventory Manager. The next two are used in interaction with Laptops, Inc. They are sent as part of an invoke request, with an

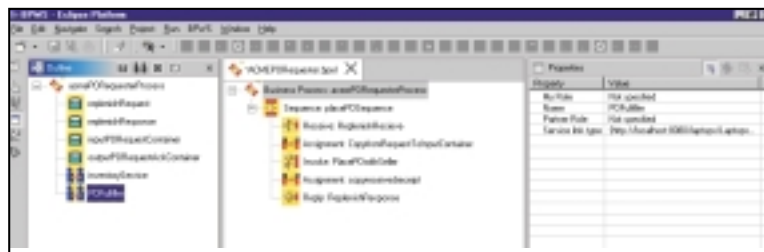


Figure 3 • Using the BPWS4J editor in Eclipse to create ACMEPORequester.bpel

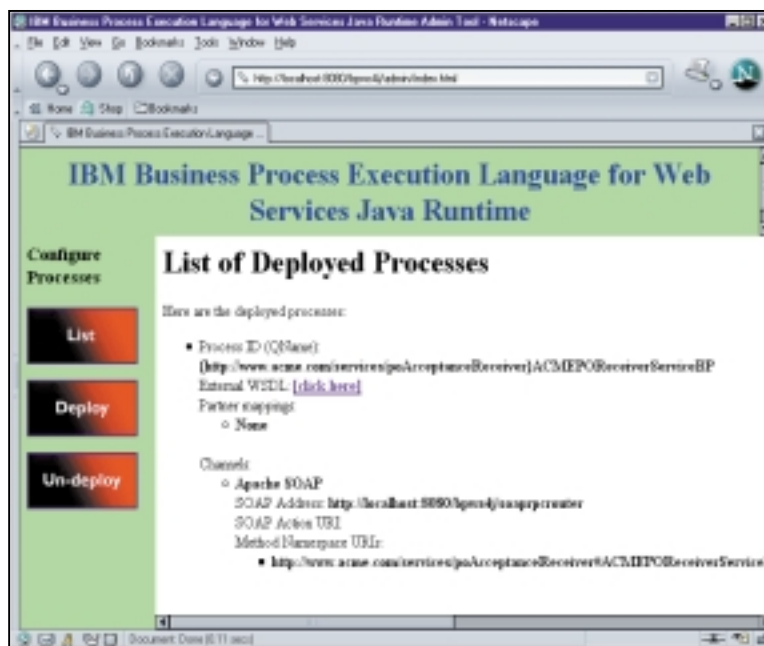


Figure 4 • Deploying a process on the BPWS4J engine

input container containing the message being sent to Laptops, Inc., and the output container containing the message sent from Laptops, Inc. (see Listing 3).

I'll now define the activity of the process. I'm using a simple sequence activity to coordinate the process; it performs the following tasks:

### Begin Sequence

- Receive a PO request from the Inventory Manager
- Assign the data from the request container to the container to be sent to Laptops, Inc.
- Invoke a "place purchase order" request with Laptops, Inc.
- Assign the information from Laptops, Inc., to the container being sent to the Inventory Manager
- Reply to the Inventory Manager with the response from Laptops, Inc.

### End Sequence

Here's how the sequence is defined:

```
<sequence name="placePOSequence">
  <receive name="ReplenishRecieve" partner="inventory
    Service" portType="acmePOR:replenishRequestPort"
    operation="replenishRequest"
    container="replenishRequest"
    createInstance="yes">
  </receive>
```

The receive activity defines the type of partners that can inter-



## What's Next?

So far, this series has covered important e-business dialogue conceptual, architectural, and construction issues. I've discussed the components of e-business dialogues and relevant standards like RosettaNet and EbXML, related these standards to Web services and BPEL4WS, and built sample WSDL and BPEL definitions. We now have executable BPEL processes that enable a simple e-business dialogue scenario using a reliable asynchronous interaction mode.

Are we there yet? Do we have a real-world Web service? Almost. We have to construct the remaining six components described in Figure 2 to enable BPEL processes to interact with internal applications and Web services. We need to start exchanging real RosettaNet-based purchase order data, and

we need to add features like correlation to the BPEL processes to make them more reliable. We also need to explore the real-world challenges faced when constructing a real-world Web service with BPEL and WSDL. Tune in next month as I expand the simple scenario created here into a robust e-business dialogue. ☛

## AUTHOR BIO

Suhayl Masud is the founder and lead consultant at Different Thinking, a consulting firm that enables organizations to conduct electronic business, by providing training, architecture, and application construction services. Suhayl's experience includes consulting as the lead technical architect for RosettaNet, where he helped define the next generation of e-business.

SUHAYL@DIFFERENTTHINKING.COM

### LISTING 1

```
<definitions targetNamespace="http://www.laptops.com/wsdl/POService"
  xmlns:LPTS="http://www.laptops.com/wsdl/POService"
  xmlns:slt="http://schemas.xmlsoap.org/ws/2002/07/service-link/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="PORequestType">
    <part name="PORequest" type="xsd:string"/>
  </message>
  <message name="ReceiptAckPORequestType">
    <part name="receipt" type="xsd:string"/>
  </message>
  <portType name="placePORequestPort">
    <operation name="placePORequest">
      <input message="LPTS:PORequestType"/>
      <output message="LPTS:ReceiptAckPORequestType"/>
    </operation>
  </portType>
  <slt:serviceLinkType name="PORequestSLT">
    <slt:role name="buyer">
      <slt:portType name="LPTS:placePORequestPort"/>
    </slt:role>
  </slt:serviceLinkType>
  <service name="placePORequestServiceBP">
  </service>
</definitions>
```

### LISTING 2

```
<definitions targetNamespace="http://www.acme.com/services/poAcceptanceReceiver"
  xmlns:ac="http://www.acme.com/services/poAcceptanceReceiver"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:slt="http://schemas.xmlsoap.org/ws/2002/07/service-link/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="POAcceptanceType">
    <part name="POAcceptance" type="xsd:string"/>
  </message>
  <message name="ReceiptAckType">
    <part name="receiptAck" type="xsd:string"/>
  </message>
  <portType name="sendPOAcceptancePort">
    <operation name="sendPOAcceptance">
      <input message="ac:POAcceptanceType"/>
      <output message="ac:ReceiptAckType"/>
    </operation>
  </portType>
  <slt:serviceLinkType name="POAcceptanceSLT">
    <slt:role name="POService">
      <slt:portType name="ac:sendPOAcceptancePort"/>
    </slt:role>
  </slt:serviceLinkType>
  <service name="ACMEPOReceiverServiceBP">
```

```
</service>
</definitions>
```

### LISTING 3

```
<containers>
  <container name="replenishRequest" messageType="acmePOR:ReplenishRequestType"/>
  <container name="replenishResponse" messageType="acmePOR:ReceiptAckType"/>
  <container name="inputPORequestContainer" messageType="laptopsPO:PORequestType"/>
  <container name="outputPORequestAckContainer"
    messageType="laptopsPO:ReceiptAckPORequestType"/>
</containers>
```

### LISTING 4

```
<process name="acmePOReceiverProcess"
  targetNamespace="http://www.acme.com/services/acmePOReceiver-Process"
  xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/"
  xmlns:acmePOA="http://www.acme.com/services/poAcceptanceReceiver"
  xmlns:laptopsPO="http://localhost:8080/laptops/LaptopsIncPlacePO.wsdl">

  <partners>
    <partner name="POAcceptanceNotifier"
      serviceLinkType="acmePOA:POAcceptanceSLT"/>
  </partners>

  <containers>
    <container name="POAcceptanceContainer"
      messageType="acmePOA:POAcceptanceType"/>
    <container name="POAcceptanceReceipt"
      messageType="acmePOA:ReceiptAckType"/>
  </containers>

  <sequence name="receivePOSequence">
    <receive name="ReceivePOAcceptance"
      partner="POAcceptanceNotifier" portType="acmePOA:sendPOAcceptancePort" operation="sendPOAcceptance"
      container="POAcceptanceContainer" createInstance="yes">
    </receive>
    <assign>
      <copy>
        <from expression="'Acme corp has received your PO Acceptance. Thank you'"/>
        <to container="POAcceptanceReceipt" part="receiptAck"/>
      </copy>
    </assign>
    <reply name="POAcceptanceReceiptAck"
      partner="POAcceptanceNotifier" portType="acmePOA:sendPOAcceptancePort" operation="sendPOAcceptance"
      container="POAcceptanceReceipt">
    </reply>
  </sequence>
</process>
```

Download the Code  
www.sys-con.com/xml





# What's New in XSLT 2.0

## A new face for XML transformations

**T**he XSLT version 1.0 language definition has been an official recommendation of the W3C since 1999. Its use has expanded dramatically in the past 18 months, for processing XML and XML/SOAP security policies and for generating HTML Web pages.

Of course, nearly as soon as the language became official, people began proposing to change it. (Indeed, the original document has a page of suggested improvements for future versions.) These efforts began as a version 1.1 proposal, which was abandoned in favor of the current Working Draft (WD). We should see XSLT 2.0 become an official W3C Recommendation sometime this year.

### Data Types

XSLT 1.0 dealt with four types of data – strings, numbers, Booleans, and nodesets. XSLT 2.0 has 48 atomic (built-in) data types, plus lists and unions constructed from them. There are now 16 numeric types; 9 variations of date, time, and duration; plus hexBinary and base64Binary, among others. Users may also create others from the built-in types to suit their needs.

Let's look at numbers. In XSLT 1.0, there was only a single variety of number, represented internally as a floating point double, and sometimes used explicitly as an integer. Now we'll have doubles, floats, various signed and unsigned integer precisions, and decimals. Decimals may be a new concept for some of you, unless you've been programming in COBOL recently. These are intended to provide exact representations of decimal fractions (e.g., dollars and cents) without the

approximation caused by using floating point. So, we now have three different kinds of numeric constants we can create, instead of one:

1. **Integer:** 1234
2. **Decimal:** 12.34
3. **Double:** 1234e-2

Notice that 12.34 used to be a double; now it's a decimal. And what about all the rest of the data types? Any type, including user-defined (derived) types, can be created using a constructor with the same name as the data type. For example:

- `xs:decimal(1234)` creates a decimal value
- `xs:date("2003-03-17")` creates a date

Having all these new types will lead to greater flexibility, and, occasionally, things to watch out for. XSLT is now a

duration, `dateTime`, `date`, `time`, `gYearMonth`, `gYear`, `gMonthDay`, `gDay`, and `gMonth`. What more could anyone want? And there will be a `format-date()` function available, although the details are not yet specified in the latest WD.

Finally, we still have strings and Booleans. But what about nodesets? Technically, they have disappeared, replaced by sequences (lists) of nodes. In general, sequences are not necessarily in document order and may contain duplicates. However, all functions from XSLT 1.0 that returned nodesets in the past now return ordered sequences with duplicates removed. Old stylesheets will still work fine.

### Path Expressions

XPath 2.0 has generalized its path expressions. Now, a primary expression (literal, function call, variable refer-

#### New XSLT 2.0 Features

- 48 data types, and optional use of schemas
- Strong type checking
- Works with XPath 2.0
- Grouping
- Regular expression matching
- Multiple inputs and outputs
- User-defined functions
- XHTML output method

#### New XPath 2.0 Features

- More general path expressions
- `if ... then ... else`
- For loops
- Casting
- Schema validation
- Node identity (`is`, `isnot`)
- Node order (`>>`, `<<`)

strongly typed language, so it's possible that a parameter to a function you call will need to be an integer – if you pass a double instead, an error will result, unless you cast it properly.

The new date and type data types are a great blessing to anyone who had to suffer through the ugly string manipulations required before. We now have

ence, or parenthesized expression) can appear at any step in the path, rather than just the first step. Basically, this allows any expression that returns a nodeset (sequence of nodes) to appear on either side of a `/`. This is especially useful for expressions like:

[Book/\(Chapter | Appendix\)/Paragraph](#)

#### AUTHOR BIO

Jeff Kenton is chairman of the OASIS XSLT/XPath Conformance Committee and the senior developer at DataPower Technology ([www.datapower.com](http://www.datapower.com)), Inc., for their XSLT 2.0 JIT compiler. Prior to joining DataPower, Jeff spent 25 years consulting in the Boston area. He was also a cofounder and the director of operating systems at Xyvision in 1982. His degrees are from MIT and Carnegie Mellon.



or

```
document("a.xml")/id("ID01")
```

There are two things to notice here. First, this applies to expressions, but not to patterns. So, you can use these in `xsl:for-each` or `xsl:value-of` statements, but not in the match patterns for templates. Match patterns have not changed, except that predicates in patterns can be XPath 2.0 expressions. The second thing to notice is that some things are legal that aren't especially useful:

```
document("a.xml")/document("b.xml")
```

or

```
anything-on-the-left/$x
```

Both of these ignore anything to the left of the last "/" and are equivalent to the nodeset that is the rightmost step. Hopefully, XSLT processors will produce a warning for these.

## Conditionals and Looping

Expressions can now include bits of flow control within them, using `if ... then ... else`:

```
<xsl:value-of select=" if(item/price
< 100) then 'cheap' else 'expensive'
" />
```

This is entirely within XPath expressions, and is separate from `xsl:if`. (Old-timers may be reminded of Algol syntax here.)

You can now generate sequences with for loops. Consider the following:

```
for $i in (0 to 9), $j in (1 to 10)
return 10*$i + $j
```

This returns the sequence of numbers from 1 to 100. Of course, in this case, a simpler equivalent would use a range expression, (1 to 100), to generate the same sequence.

## Grouping

Processing groups of related elements is a problem that comes up repeatedly on XSLT mailing lists. The required techniques are well known among experienced users, but they have to be explained anew to every HTML programmer who comes over to XSLT. Consider the sample XML shown below:

```
<cars>
```

```
<car make="Dodge" model="caravan"
color="red" price="28000" />
<car make="Ford" model="probe"
color="blue" price="14000" />
<car make="Dodge" model="caravan"
color="red" price="28000" />
<car make="Ford" model="thunderbird"
color="silver" price="45000" />
<car make="Ferrari" color="red"
price="280000" />
<car make="Dodge" model="caravan"
color="green" price="28000" />
</cars>
```

Suppose you wanted to list the cars by make, or by color. The new `xsl:for-each-group` instruction makes life much easier:

```
<xsl:for-each-group select="cars/
car" group-by="make">
Makes: <xsl:value-of select=
"current-group()/@make"
separator="," />
</xsl:for-each-group>
```

You have a lot of flexibility with the `xsl:for-each-group` element. In addition to the required `select` element shown above, the following attributes are available:

- **group-by**: allows selection of a group of elements to be treated together
- **group-adjacent**: groups of adjacent items that match
- **group-starting-with**: each node matching the pattern starts a new group
- **group-ending-with**: similarly, a matching node ends a group

Exactly one of these four attributes must appear. Within the `xsl:for-each-group` element, the new `current-group()` function gives access to the group members, and will make grouping much simpler than it was with XSLT 1.0.

## Regular Expressions

String handling in XSLT has always been a tedious process. Both XSLT and XPath have added new facilities for dealing with regular expressions. The most complex of these is `xsl:analyze-string`, which takes an input string and a regular expression. It partitions the input into a set of substrings matching the expression. These can be processed by `xsl:matching-substring` and `xsl:non-matching-substring` instructions, which can construct any content required for each of them.

For simpler regular expression processing, XPath 2.0 has added three

functions, `fn:match()`, `fn:replace()`, and `fn:tokenize()`. The first returns Boolean true or false, depending upon whether a string matches a given regular expression. The second replaces all substrings in the input string that match the regular expression with a replacement string, and returns the resulting string. The third uses a regular expression as a separator, and returns a sequence of substrings created by breaking the input at that separator. In addition, of course, XPath retains all its other string manipulation functions. This should go a long way to simplifying – and making more powerful – string handling in stylesheets.

## User-Defined Functions

User-defined stylesheet functions allow creation of functions that can be called from within XPath expressions. Declaration is simple. For example:

```
<xsl:function name="square" >
<xsl:param name="x" />
<xsl:result select="x*x" />
</xsl:function>
```

returns the square of its input parameter. Children of `xsl:function` declarations may only be `xsl:param`, `xsl:variable`, `xsl:message`, or `xsl:result`. This might seem to limit what functions are capable of doing. However, there is no limit to what can appear inside the `xsl:variable` element, including `xsl:call-template` and `xsl:apply-template`, so you really have a great deal of flexibility.

**“Having all these new types will lead to greater flexibility, and, occasionally, things to watch out for”**

## Schemas

One issue that has proven somewhat divisive in version 2.0 is the inclusion of schemas. There are those, including Microsoft and others, who believe that schemas are necessary to what people are doing with the language. Schemas provide both validation of data types in input documents, and clues to the XSLT processor about which data structures to expect. There are others who think it is too compli-

cated, and that schema validation should be a separate process. There are reasonable arguments on both sides. In the end, it was decided to define a conformance level for which schema support was optional.

**"The committees are doing a great job improving XSLT, and I expect it to be enthusiastically adopted by the XML community"**

With schema support, there is a new `xsl:import-schema` declaration. Every data type name that is not a built-in name must be defined in an imported schema. XPath expressions will be able to validate values against in-scope schemas, and will be able to use constructors and casts to imported data types.

## Inputs and Outputs

XSLT 1.0 allowed for a primary input document, auxiliary input via the `document()` function, and a single output. Version 2.0 provides for multiple inputs in several ways. There will be an `input()` function that provides access to a sequence of input nodes, and a `collection()` function that allows specification of a URI that defines a node sequence. Both of these provide access to multiple documents or document fragments. In addition, the `unparsed-text()` function will read arbitrary external resources (e.g., files) and return each one as a string.

For output, the new `xsl:result-document` instruction provides for named and unnamed output trees. Combined with `xsl:output` declarations, this allows for multiple output documents. It is a feature that has been widely requested.

## When Can I Try XSLT 2.0?

There are still issues to be resolved before the Working Drafts turn into official W3C Recommendations. Committee members have said that they hope the process will be complete by late this summer or early fall. Mean-

while, Michael Kay, who is both the editor of the XSLT 2.0 WD and the creator of Saxon, has made a version of Saxon available that supports most of the new proposal. And, of course, most suppliers of XSLT processors are working to support version 2.0 of the language as soon as it becomes official. The committees are doing a great job improving XSLT, and I expect it to be enthusiastically adopted by the XML community. 🌐

## References

- *XSLT 1.0*: [www.w3.org/TR/1999/REC-xslt-19991116](http://www.w3.org/TR/1999/REC-xslt-19991116)
- *XPath 1.0*: [www.w3.org/TR/1999/REC-xpath-19991116](http://www.w3.org/TR/1999/REC-xpath-19991116)
- *XSLT 2.0*: [www.w3.org/TR/xslt20](http://www.w3.org/TR/xslt20)
- *XPath 2.0*: [www.w3.org/TR/xpath20](http://www.w3.org/TR/xpath20)
- *XML Schema Part 1: Structures*: [www.w3.org/TR/xmlschema-1](http://www.w3.org/TR/xmlschema-1)
- *XML Schema Part 2: Datatypes*: [www.w3.org/TR/xmlschema-2](http://www.w3.org/TR/xmlschema-2)
- *Functions and Operators*: [www.w3.org/TR/xquery-operators](http://www.w3.org/TR/xquery-operators)

JKENTON@DATAPOWER.COM



## PRESENTATION

### *Database-Driven Charting Using XSLT and SVG*

Producing dynamic charts for industrial reports



## INTEGRATION

### *Integrating Data in Customer-Focused Systems*

A more complete view of your customer and competitors



## TECHNOLOGY TRENDS

### *XML and Web Service Development Tools*

Hitesh Seth continues to probe into the wealth of knowledge of the industry's top experts



## PRODUCT REVIEW

### *Microsoft BizTalk Server 2002*

An in-depth exploration

# DON'T MISS XML-J MAY

## FOCUS ON CONTENT MANAGEMENT

~continued from page 5~

of quality of service. If a particular service performs poorly, what is the root cause of the problem that is affecting the performance or reliability of that service? For example, does the service orchestration engine have a memory leak, is another service to blame, or is the problem caused by poorly written business components within our own enterprise? Or, does the problem go deeper? Perhaps the business components are performing poorly due to a problem in the application server itself, the underlying CLR or JVM (depending on whether you're a .NET-head or a J2EE-head), operating system, network, or database? You get the idea.

Quality of service cannot be measured or adjusted without a comprehensive view into the entire IT infrastructure. And yet, that view must still be relevant to the business process itself. The idea that we can alleviate our concerns about service quality solely by monitoring the SOAP messages, the service handler, or even the app server is absurd. Again, no single entity in the enterprise, from the portal to the Web service down to the lowliest

router, is an island. All, working together as seamlessly as possible, are necessary for successful business processes.

The same is true with security. For enterprises that are serious about the security of their information assets, there is no such thing as just Web services security. Rather, there is Web services security in the context of enterprise-wide security policies and mechanisms that are hopefully well-established, comprehensive, and cross-platform. Newer information sources, such as private UDDI registries, must fit into the existing security infrastructure and integrate with existing security policies, with minimal overhead, while providing the necessary protection to valuable new corporate assets such as service descriptions.

What do these three core issues of pervasive computing (quality of service, security, and delivery) mean to those of us who work with XML and XML-derived technologies? Well, besides being compelling technical and business imperatives in their own right, perhaps they can also serve as a reminder that we need to stay focused on the entire IT infrastructure even as our keen

technical interest and our fear of obsolescence compel us to concentrate on the latest standard or tool. Especially in the case of medium and large organizations, there is already a huge investment in existing infrastructure that supports the business. The new infrastructure that manages, secures, and delivers Web services must fit into the existing IT infrastructure, not the other way around. In fact, as the poet John Donne said, "No man is an Island, entire of itself; every man is a piece of the Continent, a part of the main;" and the same should be true for any new technology, including those which hold great promise for business, such as Web services and wireless. ☎

#### AUTHOR BIO

*Paul Lipton has been an architect and developer of enterprise systems for over 20 years. He is the Web services technology leader for the CA field services organization and a technology strategist in the Office of the CTO, and has represented CA in various standards organizations such as the W3C. He is a highly sought-after speaker and has also published magazine articles on many technologies including Web services and wireless.*

■ PAUL.LIPTON@CA.COM

**Ektron**  
www.ektron.com/xml



# XML-Coursebuilder, V. 1.0

## XML and XSLT format course modules into a comprehensive presentation

**A**s a corporate trainer of Internet technologies, I've often run into situations where a company will look at a set of courses within a curriculum and say, "I want Section 2 of that course, followed by Section 4 from this other course, and then we've got a real-world project we would like you to review with the class at the conclusion of the course."

It could take hours to cut and paste material from the various courses and build a presentation based upon those requests. After nights of drudgery and muttering under my breath, I decided to develop a simple client-side XML application that could handle all the work for me.

### Project Overview

XML-Coursebuilder uses XML and XSLT to format modules of course material into a comprehensive presentation. The stylesheets build a table of contents, consecutively named screen displays, summary screens, and a full navigation system on the fly, in the browser. The course material is modularized so that individual sections of a course can be included in the final output by using a simple call to an XML module.

My company has an XML curriculum of courses that includes:

- Introduction to XML
- XSLT: Transforming and Formatting XML
- XML Application Development: 7 Projects
- VoiceXML: 10 Projects to Voice Enable Your Website

I can build a four-day presentation with XML-Coursebuilder by selecting various modules from the courses, cre-

ating the presentation in the time it takes to type calls to the various selected modules. Content, navigation, table of contents, and summaries are all included in the final output. The course is displayed on the fly, in a browser, with no external compilation. Because most of my clients use a corporate intranet, I can assume the training machines are set up with MSIE 6+, which has a built-in XML/XSLT processor.

Taking this a step farther, I have used XML-Coursebuilder to include multiple author content within a presentation. A group of instructors can modularize all of their courseware and store it in a central location for access by the application, and each of their modules can be used as a section in the new presentation. Since the presentations are built on the fly from original sources, all courseware has the most current material available. I have used this concept to build technology overview courses by pulling in just the introductory modules from the courses the client was interested in.

### File Structure

The XML-Coursebuilder files have been modularized so that changes to formatting are isolated from the content. Each XML file holds one complete section of data in a course. A section can contain an unlimited number of pages and a course can contain an unlimited number of sections.

Each XML file places a call to an XSL stylesheet that outputs HTML formatting. The generated HTML pages include calls to an external JavaScript file for creating the navigation system and an external CSS file for setting the properties of the HTML pages (see Figure 1). All of the formatting files are

located in a single "coursebuilder" directory, allowing multiple presentations to access the same XML-Coursebuilder files (see Figure 2). This makes it simple to apply global changes to the layout of your courseware, presenting a consistent look across all of your presentations.

### File Manifest

Following is a high-level overview of each of the core files within the XML-Coursebuilder system. You can download the complete set of files for experimenting at [www.TechTrainingWorkshop.com/coursebuilder](http://www.TechTrainingWorkshop.com/coursebuilder) as well as at [www.sys-con.com/xml/sourcec.cfm](http://www.sys-con.com/xml/sourcec.cfm).

#### clientSidePagination.xsl

The clientSidePagination.xsl file contains XSLT templates for transforming the content from the XML files. It starts with a root-level Master Page template that creates an HTML wrapper for each individual page (see Listing 1). Next, a "navigation" template (see Listing 2) generates a navigation system at the bottom of each HTML page, displaying "Back" and "Next" buttons, along with linked page numbers to every page within the section (see Figure 3). (For detailed discussion of using JavaScript and XSLT to build dynamic navigation systems, see [www.bayes.co.uk/xml/](http://www.bayes.co.uk/xml/).) The format of each page is handled with the "page" template and the file ends with an alphabetical listing of each of the templates that will handle elements from the XML input (see download files).

#### clientSideToc.xsl

The clientSideToc.xsl file pulls in the names of the XML files that are used to build each section. The code shown in Listing 3 creates a link to each XML file used in the presentation.

### AUTHOR BIO

Mark Miller has been a corporate trainer of Internet technologies since 1997 and has taught at Hewlett-Packard, Autodesk, SGI, and Charles Schwab. Mark's new project, "Experts Online: Workshops from the Trenches," will incorporate live, Internet-delivered training with industry experts showing hands-on projects in real time. He is the author of VoiceXML: 10 Projects to Voice Enable Your Website by Wiley Publications.

### clientSideSummary.xml

This file does basically the same as the table of contents, formatting it so that all pages, not just section titles, can be listed as part of the summary.

### clientSidePagination.js

The JavaScript navigation system used by each of the output HTML pages is stored in clientSidePagination.js. Complete documentation for creating this type of navigation can be found at [www.bayes.co.uk/xml](http://www.bayes.co.uk/xml).

### clientSidePagination.css

For a consistent look and feel across all presentation materials, clientSidePagination.css is used. Each of the dynamically generated HTML pages places a call to this file. Any HTML formatting of output should be done here.

### Data Description (\*.xml)

I created a simple element set based upon what I use during most of my presentations. The root element is called <course>, which holds <topic> elements for each of the sections to be displayed. Each <topic> contains a <title> for the chapter and then multiple <page> elements. Pages are broken down into <talkingPoints> that consist of a <heading> plus <outline> tags holding <item> elements (see Listing 4). Links, images, and paragraphs can be inserted with various other tags.

### Final Output (\*.html)

The final HTML output is shown in Figure 4. The table of contents in the background is used to open up a new window when a section is chosen. When the section is completed, the window is closed, taking the user back to the table of contents. The final output includes a table of contents, a set of HTML pages for each of the sections, a summary for each of the sections, and a final summary for the complete course.

### Putting the Pieces Together

A sample course is included with the download for this article. The processing files are stored in the "coursebuilder" directory, while the sample course is stored in the documentation directory. In order to build the course dynamically, you must have MSIE 6+ installed on your machine or some other client-side process to display the output in your browser. Open toc.xml in the browser to view the HTML pages.

To build your own presentation, create a new directory and save a copy of toc.xml into that directory. Rename the

XML files to point to the XML files that you'll create. You can use one of the XML files in the documentation directory as a template for building your own application. Add as many XML modules to your presentation as you'd like.

### Conclusion

The system shown here is a simple one that has powerful implications. Isolating the data into separate modules for easy access and reuse can work wonders for presentation building. I had a client who called to describe what she thought was a unique request for courseware to present to her company. Using XML-Coursebuilder, I was able to construct and ship her a prototype presentation within ten minutes.

My favorite idea is to compensate authors for their work on a per-module basis instead of having to write a complete course from scratch. Taking that a step farther, a group of authors could use RSS to make summaries available from their sites. Anyone needing a specific module type could generate a list of summaries and select those that are most appropriate. This would allow experts in all fields to generate modules and make them available to courseware developers.

Picture a large, distributed repository of modules with the most current infor-

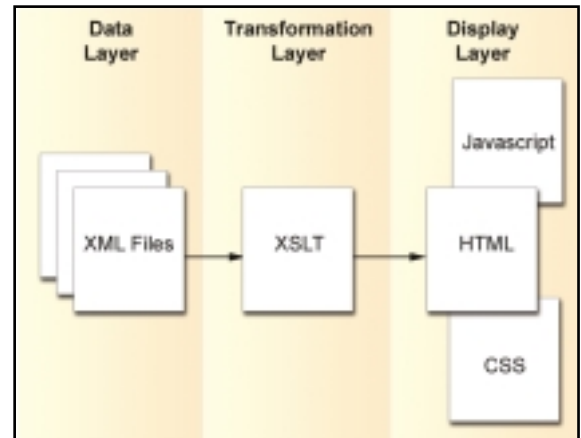


Figure 1 • System architecture



Figure 2 • Directory structure



Figure 3 • Bottom navigation

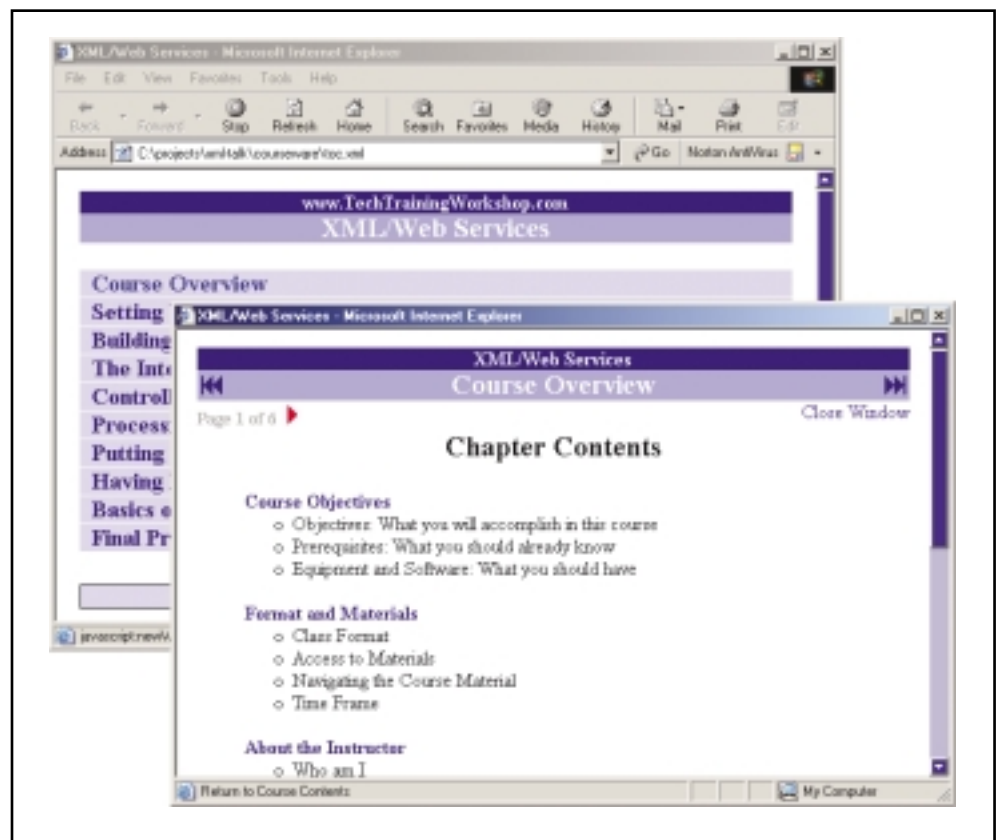


Figure 4 • Final output



mation available from the people who are working in the trenches. I would access a module by Jeni Tennison or Michael Kay without hesitation, or use an XML-ized version of Dave Pawson's FAQs for a starting point of tricks and traps with XSLT. However, unheralded but well-deserving course authors could also get exposure by participating in the construction of the repository.

If you would like to participate in the continued development of this set of XSLT files and general development of the XML-Coursebuilder concept, e-mail to [coursebuilder@techtrainingworkshop.com](mailto:coursebuilder@techtrainingworkshop.com).

### Resources

- Download project files: [www.techtrainingworkshop.com/coursebuilder](http://www.techtrainingworkshop.com/coursebuilder)

- Kay, Michael. (2000). *XSLT Programmer's Reference, 2nd Edition*. Wrox Press.
- RSS: <http://backend.userland.com/rss>
- XSLT/Javascript Navigation: [www.bayes.co.uk/xml](http://www.bayes.co.uk/xml)
- XSLT Mailing List: [www.mulberrytech.com/xsl/xsl-list](http://www.mulberrytech.com/xsl/xsl-list)

MARK.MILLER@TECHTRAININGWORKSHOP.COM

#### LISTING 1

```
<?xml version="1.0"?>

<!--
  Author: Mark Miller
  Contact: mark.miller@techtrainingworkshop.com
  Basic pagination script: http://www.bayes.co.uk/xml/
  Date: March 2003
  Note: Please consider sending me any changes you make to
  the file so that I can include them in future updates.
-->

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:output method="html" indent="no"/>

<xsl:param name="pagenumber" select="1"/>

<!-- ***** MASTER PAGE FORMAT ***** -->
<xsl:template match="/">
<html>
  <head>
    <title><xsl:value-of select="topic/metadata/
      presentation/event"/></title>

    <!-- PATHS SHOULD BE RELATIVE TO THE XML FILE,
      NOT THE XSL FILE -->
    <script language="javascript"
      src="../resources/clientSidePagination.js"/>
    <link rel="stylesheet" type="text/css"
      href="../resources/clientSidePagination.css"/>
  </head>

  <body onLoad="window.status='{topic/title}'; return
    true;">

    <!-- SELECT THE PAGES -->
    <xsl:apply-templates select="topic/page
      [{pagenumber}]">
      <xsl:with-param name="element" select="'page'" />
      <xsl:with-param name="pagenumber"
        select="$pagenumber" />
    </xsl:apply-templates>

    <!-- NAVIGATION -->
    <xsl:call-template name="navigation">
      <xsl:with-param name="element" select="'page'" />
      <xsl:with-param name="pagenumber"
        select="$pagenumber" />
    </xsl:call-template>

    <!-- COPYRIGHT -->
    <xsl:apply-templates select="topic"/>
  </body>
</html>
</xsl:template>

...

</xsl:stylesheet>
```

#### LISTING 2

```
<?xml version="1.0"?>

<!--
```

```
Author: Mark Miller
Contact: mark.miller@techtrainingworkshop.com
Basic pagination script: http://www.bayes.co.uk/xml/
Date: March 2003
Note: Please consider sending me any changes you make to
the file so that I can include them in future updates.
-->
```

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:output method="html" indent="no"/>

<xsl:param name="pagenumber" select="1"/>

<!-- ***** MASTER PAGE FORMAT ***** -->
<xsl:template match="/">
  ...
</xsl:template>

<!-- **** NAVIGATION: BOTTOM OF PAGE **** -->

<xsl:template name="navigation">
  <xsl:param name="element" />
  <xsl:param name="pagenumber" />
  <xsl:variable name="total" select="count(topic/*[name()
    = $element])" />

  <!-- UNCOMMENT FOR TESTING
    Page Number: <xsl:value-of
      select="$pagenumber" /><br/>
    Element Name: <xsl:value-of select="$element" /><br/>
    Total number of pages in this topic: <xsl:value-of
      select="$total"/>
  -->

  <!-- RESTRICT DISPLAY TO 600 PIXELS -->
  <table width="600" border="0" align="center">
    <tr>
      <td align="right">
        <a href="javascript:window.close();">Close
          Window</a>
      </td>
    </tr>

    <tr>
      <td>
        <div class="navbar">
          <table border="0" width="600" cellpadding="0"
            cellspacing="0" class="tableColor">
            <tr>

              <!-- BACK BUTTON -->
              <td align="left" width="10%">
                <xsl:if test="$pagenumber > 1">
                  <a onMouseOver="navigationInfo('Go back one
                    page in this section'); return true;"
                    onMouseOut="showPageInfo('Currently on
                    Page {pagenumber}'); return true;"
                    href="JavaScript:changePage({pagenumber
                    -1});">
                    <span class="arrowsBottom">3</span><span
                      class="directionWord">Back</span></a>
                </xsl:if><br/>
              </td>

              <!-- PAGE NUMBERS WITH LINKS -->
              <td align="center"><span
```

```

        class="navbarText">Jump to Page: </span>
<xsl:for-each select="topic/*[name() =
$element]">

    <xsl:variable name="currentNumber"
        select="count(preceding-sibling::page)+1"/>

    <xsl:choose>
        <xsl:when test="not($currentNumber =
            $pagenumber)">
            &#160;<a onmouseover="navigationIn
                fo('Page {$currentNumber}:
                {title}'); return true;"

onMouseOut="showPageInfo('Currently on Page {$pagenumber}');
return true;"

href="JavaScript:changePage({$currentNumber});">
                <xsl:value-of select=
                    "$currentNumber" /></a>

            </xsl:when>
            <xsl:otherwise>
                &#160;<span
                    class="currentPage"><xsl:text>
                        </xsl:text><xsl:value-of
                            select="$currentNumber" /><xsl:text>
                        </xsl:text></span>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:for-each>
    </td>

    <!-- NEXT BUTTON -->
    <td align="right" width="10%">
    <xsl:if test="$pagenumber <= $total">
        <a onmouseover="showPageInfo('Currently on
            Page {$pagenumber}'); return true;"
            onmouseover="navigationInfo('Go to the
            next page in this section'); return
            true;"
            href="JavaScript:changePage
                ({$pagenumber +1});">
            <span class="directionWord">
                Next</span>
            <span class="arrowsBottom">4</span></a>
    </xsl:if><br/>
    </td>

    <!-- CLOSE HTML TABLE -->
    </tr>
</table>
</div>

</td></tr></table>

</xsl:template>
...
</stylesheet>

```

### LISTING 3

```

<!-- XML file calling each of the presentation modules -->
<?xml version="1.0" standalone="no"?>
<?xml-stylesheet type="text/xsl"
href=" ../coursebuilder/clientSideToc.xsl"?>

<course>
    <topic href="introduction.xml"/>
    <topic href="creating-xml.xml"/>
    <topic href="defining-dtd.xml"/>
    <topic href="output-xslt.xml"/>
    &footer;
</course>

<!-- XSL Snippet to Build TOC links-->
<xsl:for-each select="topic">
    <div class="tocChapterTitle">
        <a href="javascript:newWindow('{@href}')">
            <xsl:value-of
                select="document(@href)//topic/title"/>

```

```

        </a>
    </div>
</xsl:for-each>

```

### LISTING 4

```

<?xml version="1.0" standalone="no"?>
<?xml-stylesheet type="text/xsl"
    href=" ../coursebuilder/clientSidePagination.xsl"?>

<!DOCTYPE topic [
    <!ENTITY footer SYSTEM "metadata.xml">
    <!ENTITY placebo ">
]>

<topic subject="xsl">

    <!-- METADATA INFORMATION ABOUT THIS SESSION -->
    &footer;

    <!-- TITLE OF THIS SESSION -->
    <title>Course Overview</title>

    <!-- ***** PAGE ***** -->
    <page subject="contents">
        <title>Chapter Contents</title>
        <heading>Introduction</heading>
    </page>

    <page subject="intro-materials">
        <title>Format and Materials</title>
        <talkingPoint>
            <heading>Class Format</heading>
            <outline>
                <item>Short lecture and discussion of a relevant
                    point</item>
                <item>Hands-on lab to practice utilizing the
                    point</item>
                <item>Finished solution for each lab</item>
            </outline>
        </talkingPoint>
        <talkingPoint>
            <heading>Access to Materials</heading>
            <outline>
                <item>All materials are online</item>
                <item>Code samples, resources and practice tools
                    <outline>
                        <item>download class <inlineLink href="code.exe">code
                            samples</inlineLink></item>
                        <item>save the file to your desktop</item>
                        <item>double click the 'code.exe' file that is stored
                            on the desktop</item>
                        <item>choose 'c/' as the location for 'Unzip to
                            folder' field</item>
                        <item>choose 'Unzip'</item>
                    </outline>
                </item>
            </outline>
        </talkingPoint>
    </page>
    <page subject="intro-aboutme">

        <title>About the Instructor</title>

        <talkingPoint>
            <heading>Who am I</heading>
            <outline>
                <item>Mark Miller</item>
                <item>Corporate Trainer of Internet
                    Technologies</item>
                <item>NYC, NY</item>
                <item>mark.miller@TechTrainingWorkshop.com</item>
            </outline>
        </talkingPoint>

        <image>
            <src>../images/mark-vermont.jpg</src>
            <width>300</width>
            <height>217</height>
            <border>1</border>
            <alt>Stowe, Vermont... highest point in the state</alt>
        </image>
    </page>
</topic>

```



# Struts and XSLT – It's Not an Either/Or Decision

WRITTEN BY  
FRANK NEUGEBAUER

*Leverage the strengths of both Struts and XSLT to  
create dynamic, configurable Web applications*

**W**hen developing Web applications that use Java and XML there are many options, including (among others) the Apache Struts framework and the Extensible Stylesheet Language Transformation (XSLT) language.

At first blush, these options may seem like an “either/or” proposition, considering the fact that the view portion of Struts serves essentially the same purpose as XSLT: to render the view, or visual output, of the application. This article will show how XSLT can be used within the view layer of Struts to leverage the strengths of both and allow you maximum flexibility in the visual presentation of your Java Web applications.

Some of you may be thinking design patterns at this point. That is, Struts uses the model 2 (M2) design pattern, which began as the model-view-controller pattern (see *Design Patterns: Elements of Reusable Object-Oriented Software* from Addison Wesley). The M2 pattern has been extended by incorporating XSLT (for XML applications), creating the model 2X (M2X) pattern, which is very close to what I'm about to show you. However, I've taken yet another deviation, which I believe adds a greater level of flexibility and robustness.

For the sake of brevity I'm assuming at least introductory knowledge of both XSLT and Struts. I'm also assuming some basic knowledge of the Cascading Stylesheet (CSS) language, because what good would a flexible view layer be without an externalized look-and-feel? I will provide a brief overview of both Struts and XSLT, but those overviews will be neither extensive nor adequate to fully understand the contents of this article.

## Overview

One of Stephen R. Covey's *Seven Habits of Highly Effective People* (Simon & Schuster, 1990) is to “begin with the end in mind.” In keeping with Mr. Covey's habit, allow me to show you what we're trying to accomplish. After I show you the end, I'll put everything together piece by piece, to (hopefully) solidify these concepts.

Figure 1 shows each part of the application, highlighting the Struts and XSLT parts.

Taking a step back from Figure 1, Figure 2 shows which parts of Figure 1 belong to model, view, and controller, respectively.

What I've added to Struts is a generic view bean, which uses XSLT to transform XML into the view output (e.g., XHTML). For any given view to be rendered, any XML and any XSL stylesheet can be used together to generate the output. The rationale behind this usage, with Web pages in particular, comes from the observation that many (if not most) display (as opposed to input) Web pages are similar in nature.

Generally speaking, Web applications have a kind of template for visually appealing aspects such as a navigation and/or status bar (with and/or without graphics). Within this template, there are usually one or more sections that contain important data; that is, business data within HTML. Take Figure 3, for example.

The banner and navigation portions of Figure 3, along with the look-and-feel defined by the colors and fonts, make up the template I mentioned earlier. Within this template, there is a listing of persons, ordered by last name. This listing constitutes the dynamic, or business, data, which (in this case) is represented within an XHTML table.

If using Struts alone, this same output could be generated, there's no doubt about that. However, more Struts tags would be required, and they could become cumbersome (HTML tables and Struts tags are a bit frustrating sometimes). Furthermore, performing XPath operations such as selecting certain nodes in the set, or XSLT functions such as sorting, is not nearly as simple using Java as XPath and XSLT. If that's not enough to convince you, consider XSLT (the stylesheet language) as a language that is specialized for turning XML into something else, such as XHTML. (Java isn't specialized in this way, so Struts isn't specialized in this way either.)

Within the pattern I'm describing, the data portion of the page is presented using XSLT, because XSLT is great at taking XML data and transforming it into XHTML (and in this case, sorting by last name). The rest of the visual presentation is plain XHTML, which is created as a JSP. Using this pattern leverages the programming robustness of XSLT while at the same time allowing the graphic design team to use a JSP, which can be

designed with a GUI tool. In other words, this framework allows programmers and graphics/Web design folks to do what they do best.

## Struts

Apache Struts is an open-source Java/JSP Web application framework available as part of the Apache Software Foundation's Jakarta project. Struts provides many of the parts (or beginnings of the parts) shown in Figures 1 and 2. Specifically, Struts provides the ActionServlet, ActionForm (which is subclassed), and DynaActionForm (with Struts 1.1) classes. These classes understand how to take HTML form data and pass it along to the necessary classes in order to enable business processing, which is contained within a subclass of Struts' Action class. Your subclass of Action performs the necessary business processing and forwards the reply to a JSP along with a Java bean containing the response data. This response data is displayed within the JSP using special Struts JSP tags. Struts understands how to go from page to page by utilizing a Struts-specific XML file, named `struts.config.xml`, which is read by the ActionServlet (see Figure 1).

But that's not all Struts provides. It also has robust error handling, and something called "Tiles" (Struts 1.1, the same basic concept is called "templates" in 1.0x), which enable you to break down a page into sections (a lot like I did in Figure 3) and use different Struts interactions (e.g., Actions) to paint each section in a reusable fashion. However, for simplicity I've left most of the error handling and all of the Tiles out of the example I'm going to present.

Without Struts, you'd most likely create something very similar – I've done so before Struts came along, and have seen at least one other person I know do the same. That's one of the reasons Struts is so popular; it provides a framework to do what you'd likely do anyway when creating a Web application. Of course, another reason Struts is so popular is that it's free, including source code.

## XSLT

XSLT is a World Wide Web Consortium (W3C) recommendation that is intended to provide a way to transform XML into another form. The most common form is XML transformations (with XML to XHTML being an example of this form). The W3C's recommendation has been implemented by several companies/organizations, including Microsoft (XSLT is built into Internet Explorer version 4 and higher), and the Apache Software Foundation's Xalan project, which I'll be using for the example presented shortly.

XSL transformations consist of four basic parts: an XML source document, an XSL stylesheet (which is an XML document itself), an XSL processor, and a destination document. The XML source document and XSL stylesheet are read by the processor, which then performs the transformation producing the destination document (I deliberately didn't use the term "destination XML document" because technically, XSLT can transform XML into any other textual representation).

The most important part of XSLT, from the visual programmer's perspective, is the XSL stylesheet. The stylesheet contains the code that tells the processor how to transform the source to the destination. This XSL code uses templates, which catch desired XML from the source, and performs processing on it. How it does so is a "simple" matter of details, which are outside the scope of this document. If you're new to XSLT, there are many excellent resources on the Web that can assist you as you learn XSLT.

XSLT is a very robust language that can transform XML into anything ranging from HTML to plain text. Having a firm grasp

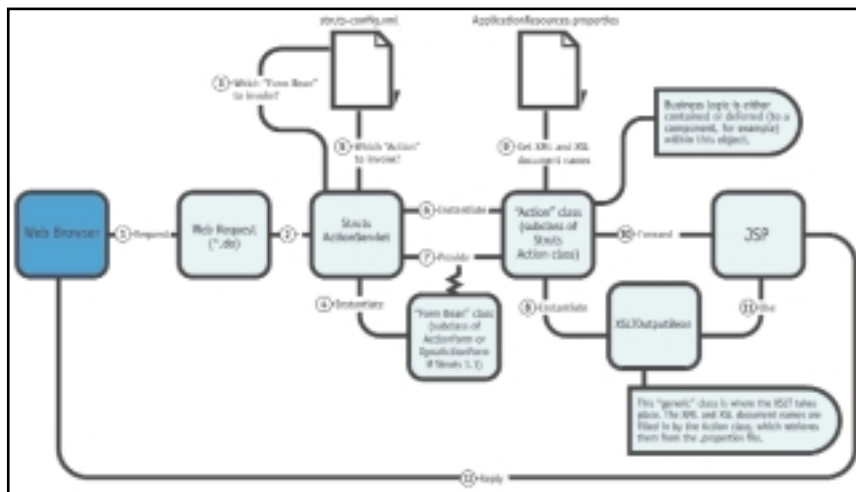


Figure 1 • The flow through an application using Struts and XSLT together

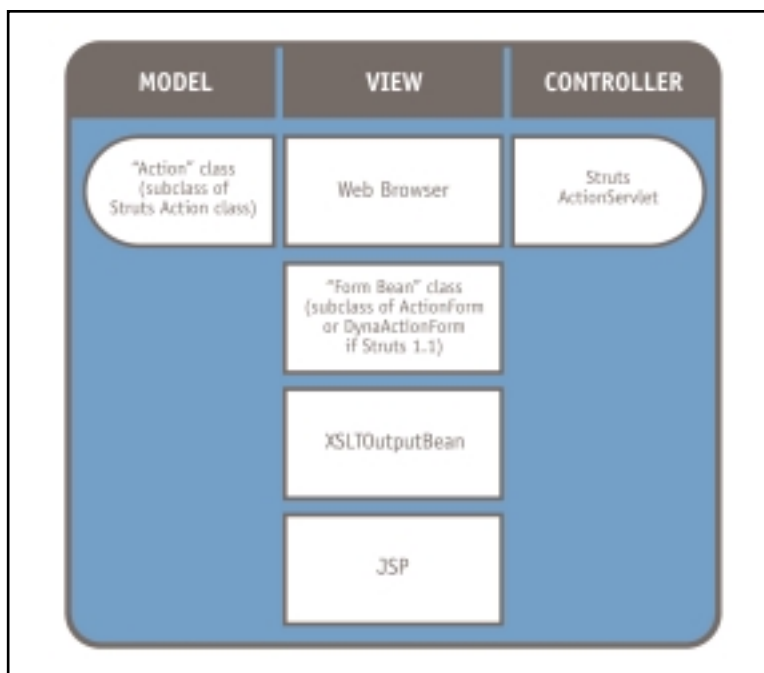


Figure 2 • Figure 1, broken apart into model, view, and controller parts

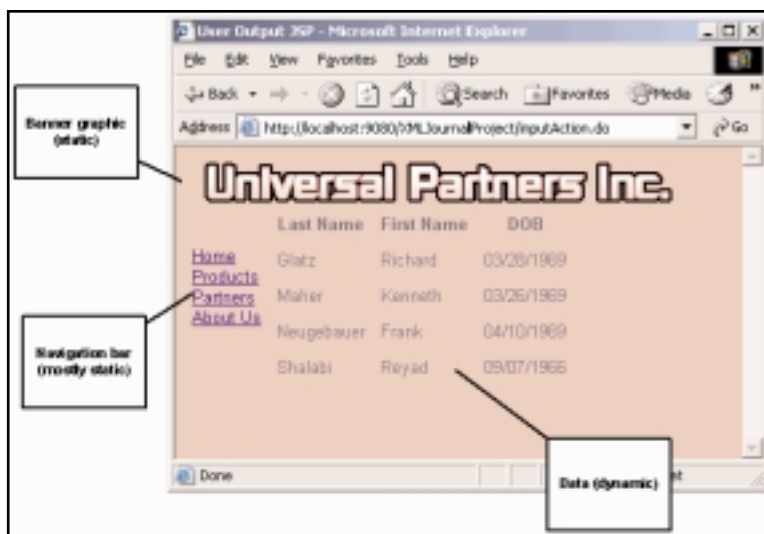


Figure 3 • Anatomy of a display Web page, like many others





of XSLT is a must for any XML programmer. Within this article, I use XSLT to leverage its strong XML-to-HTML characteristics, while doing so within Struts, which has strong Web application framework characteristics.

## Sample Application

Enough of this over-arching generic stuff! It's time for me to move on to the sample. The sample application that follows is a basic user registry, which contains a simple HTML form page and a single output page. The data is stored within an XML file that is updated whenever a new user is added. In other words, the input page provides data that gets inserted into the proper place within an XML file that is displayed after the update.

### The HTML input page

The easiest portion of the sample to create is the HTML input page. I could have created the input page as a JSP, but for simplicity I've elected to create it using plain HTML. To externalize the look-and-feel of the application (e.g., colors, font face), I used an external cascading stylesheet file (named lookandfeel.css) that has entries similar to the following snippet.

```
BODY
{
    BACKGROUND-COLOR: #d7a8a9;
    COLOR: #333366;
    FONT-FAMILY: 'Default Sans-Serif'
}
```

This snippet sets key look-and-feel elements for the HTML BODY tag, including basic color and font settings. Similar entries exist for other tags (e.g., H1, H2, and TD), which, when taken as a whole, define the entire look-and-feel for the Web application. The best part of using this stylesheet is that if I want the look-and-feel to change for every page that uses the CSS, I only have to change the stylesheet, not the pages that use it (such functionality is a basic benefit of externalizing a stylesheet). Perhaps I don't need to state that the CSS is entirely different from the XSL stylesheet, but I'll mention it just to be safe.

Figure 4 shows what the inputPage.html file looks like when rendered within a Web browser.

### The Struts ActionForm and Action subclasses

Obviously, some code has to handle the HTML form, and this is where the first Struts parts come into the application. To retrieve and store the HTML form data, I created a Java class named InputPageForm, which extends the Struts ActionForm class. Struts semi-automatically maps the HTML form data with the data in the Java class. I created three private class variables in the class, which correspond to the three HTML form input elements, and named them the same as the "name" values in the HTML form.

```
private String firstName = null;
private String lastName = null;
private String dob = null;
```

I then added getters and setters for the variables. Struts takes care of the rest at runtime (by using the struts-config.xml mapping, which I'll show shortly).

After creating the ActionForm, I created an Action class (a subclass of the Struts Action class), named InputActionAction, which performs the business logic. At the heart of the Action

class is the execute() method, where processing starts.

In the case of this sample, the Action class's execute() method first uses a Struts utility class known as PropertyUtils to retrieve the data from the ActionForm without actually knowing the Java type explicitly (Struts knows the actual type because it's defined within struts-config.xml). That code is simple, and is shown in the snippet below.

```
// Get form values
String lastName =
(String)PropertyUtils.getSimpleProperty(form, "lastName");
String firstName =
(String)PropertyUtils.getSimpleProperty(form, "firstName");
String dob = (String)PropertyUtils.getSimpleProperty(form,
"DOB");
```

The form object, the first parameter to the getSimpleProperty() method, is passed by Struts into the execute() method of the Action class automatically. At this point, I added an entry within struts-config.xml, which maps the form to the proper Java class.

```
<form-beans>
  <form-bean name="inputPageForm">
    type="com.neuggsville.Forms.InputPageForm">
  </form-bean>
</form-beans>
```

After the data has been retrieved, the Action class retrieves the contents of an XML file (named "users.xml"), then uses the Apache Xerces package to modify the XML within a method called createNewUser(). The createNewUser() method adds the new user in the appropriate location, using the data from the ActionForm class, then saves the XML document. Obviously, your business logic and persistence mechanism are likely to be vastly different from my simple example. However, the structure would remain essentially the same. Listing 1 shows the contents of users.xml.

### The view bean, XSL stylesheet, and output JSP

At this point I set my InputActionAction class aside so I could create the view bean, XSL stylesheet, and output JSP. I'll return to the InputActionAction class after the output view artifacts have been created.

The view bean, named XSLTOutputBean, does not inherit from any Struts class, and recall from Figure 1 that this view bean has been XSLT enabled. The class itself is very simple in that it has only three data members, as shown in the code snippet below.

```
private String theXML;
private String theXSL;
private String theResult;
```

The XML and XSL Strings represent paths to the XML and XSL documents, respectively (these values are set within the Action class, as I'll demonstrate before too long). The result String is the actual result of the XSLT transformation, and is represented as XHTML. The main processing of this class happens within the getTheResult() method, which is more than a simple getter method to retrieve the "theResult" data element. The method uses classes from the Xalan package to perform an XSL transformation on the XML using the given XSL stylesheet. See Listing 2 to see the entire contents of the getTheResult() method.

Speaking of the XSL stylesheet, I could have created the stylesheet the moment the XML data and final output HTML



screen were defined. Thinking about the order of events in larger projects, being able to have a sample of the end data early on allows graphic and front-end programmers (e.g., XSL stylesheet writers) the opportunity to develop the entire front end in tandem with the Action class. Such parallel development is crucial to the success of large projects and a real advantage to using XSLT and Struts in this way.

Programmatically speaking, the XSL stylesheet (named `usersOutput.xsl`) is quite simple. It has two templates, one to match the root “Users” element and another to match the “User” element child (or children) nodes.

The “Users” template creates the XHTML table structure, then calls the “User” template, “passing” (not in the Java sense, but in the XSL sense) it the “User” nodes, sorted by “LastName.” Listing 3 shows the “Users” template.

Most of this listing is just XHTML, which creates a simple table. However, the XSL processor will only run this code when a “Users” element is caught in the source XML document. Then, after creating an HTML header, the XSL element `<xsl:apply-templates select="User">` “selects” (this is an XPath expression) each “User” element under a particular “Users” element, for processing by a “User” template (which also appears in the stylesheet). Then, prior to deferring to the “User” template, the `<xsl:sort>` element performs a simple sort by the “LastName” element.

The “User” template simply outputs each child of the “User” node (i.e., “LastName”, “FirstName”, and “DOB”) as columns in the HTML table. The contents of this template are contained below.

```
<xsl:template match="User">
  <TR>
    <TD><xsl:value-of select="LastName" /></TD>
    <TD><xsl:value-of select="FirstName" /></TD>
    <TD><xsl:value-of select="DOB" /></TD>
  </TR>
</xsl:template>
```

This simple code outputs the value of the LastName, FirstName, and DOB elements as rows in an XHTML table.

Recall that the result of the XSLT is going to be placed within the context of a larger XHTML document, which is defined within the output JSP. That JSP uses the `XSLTOutputBean`’s `getResult()` method to paint the XHTML table labeled “Data (dynamic)” in Figure 3. The output JSP, named `userOutput.jsp`, is mostly plain XHTML (the template I mentioned earlier), with a very small Struts tag, which gets the result from the `XSLTOutputBean`.

```
<bean:write
  name="theBean"
  property="theResult"
  filter="false"
/>
```

The `<bean>` JSP tag is provided by Struts and identifies the servlet session variable (i.e., “theBean”) and data element (i.e., “theResult”) to use when executing the JSP. Again, such coding would have to be done without Struts, and it wouldn’t be as clean and concise, unless you built your own Struts-like framework.

To maintain the same look-and-feel, I used the same CSS file within this JSP.

### Finishing up the Action subclass

At this point, I have two options. I can edit the `struts-config.xml` file, creating the mapping to the Action and JSP forward, or I can continue coding the `InputAction` class. I’m going to finish the Java coding before editing the `struts-config.xml` file,

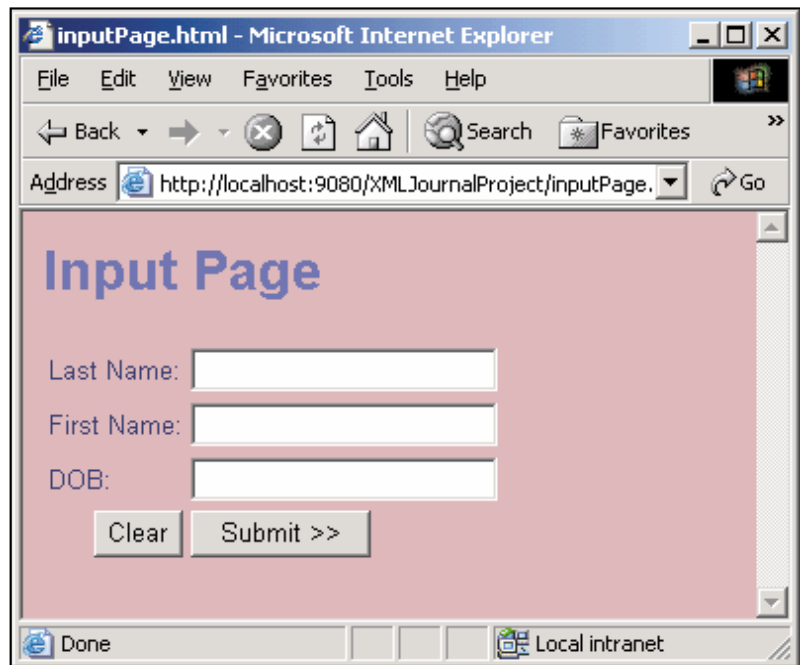


Figure 4 • The HTML input page

for no particular reason.

The only things left to do with the Action class are to instantiate the view bean and forward the result to the JSP. The names of the XML document and XSL stylesheet are contained within an externalized property resource bundle (named `ApplicationResources.properties`). The files are retrieved using HTTP paths, but you have the option to implement the retrieval in a number of different ways (see the Apache Xalan `StreamSource` class to learn about your options).

After the view bean has been instantiated and initialized, a Struts class named `ActionMapping` is used to forward processing to the view JSP through a mapping that exists within the `struts-config.xml` file that maps a simple String (“output”) to the actual JSP (you’ll see this mapping in the next section).

At this point, all of the development artifacts are complete. Next, it’s time to play connect the dots using the `struts-config.xml` file.

### Creating the flow using struts-config.xml

The mappings for the Action class and JSP forward are contained within the `struts-config.xml` file. The mappings are used by the `ActionServlet` to point processing in the right direction at runtime. As an XML file, the flow then becomes external to the application (a highlight of using Struts). Listing 4 contains those mappings.

For those of you familiar with Struts, you’ll notice that I made some simplifying compromises to my sample application. I didn’t use Struts validation, internationalization, or extensive error-handling capabilities. I also didn’t leverage the Struts 1.1 `DynaActionForm`, which allows for the definition of the HTML form within `struts-config.xml`. Again, I made these compromises for the sake of simplicity, not because such functionality isn’t entirely useful (it is!).

### Some Words of Caution

It’s important to note that although using XML and XSLT in the way I described within this article is very robust and flexible, this solution isn’t right for every problem. First off, not all Web applications use XML as a messaging mechanism (and some



necessarily shouldn't use XML). Second, and perhaps more important, even if XML is the messaging mechanism, XSLT may not scale well if the XML source document(s) are arbitrarily large and a document object model (DOM) is used as in the example. In situations where the source XML may be very large, be sure to perform thorough performance and scalability testing on your application (as early on as possible, just in case it doesn't scale or perform).

## Conclusion

If you run the application and enter data in the HTML form (Figure 4), you'll get a result that looks like Figure 3. In a traditional application, there are many interactions, some of which accept HTML form data, some of which display data. In the case of display pages in particular, using XSLT within the Struts framework allows you to not only leverage the right tool for the job (XSLT for display of XML data, Struts for application framework services), but allows for the right persons for those jobs as well. I hope you find this Struts/XSLT pattern useful, if not "as-is," then as some derivation.

## Acknowledgement

Special thanks to my good friend Richard Glatz, a talented graphic artist who created Figures 1 and 2 along with the banner

graphic for Figure 3. Working with Rick has shown me how critical the relationship between artists and programmers is, and that great Web sites require both talents in equal measure. ☛

## Resources

- *The Apache Struts Framework*: <http://jakarta.apache.org/struts/index.html>
- *XSLT*: [www.w3c.org/TR/xslt](http://www.w3c.org/TR/xslt)
- *CSS*: [www.w3c.org/Style/CSS](http://www.w3c.org/Style/CSS)
- *Apache's Jakarta project*: <http://jakarta.apache.org/struts/index.html>
- *Apache's Xalan project*: <http://xml.apache.org/xalan-j/index.html>
- *XPath*: [www.w3c.org/TR/xpath](http://www.w3c.org/TR/xpath)
- *Apache Xerces*: <http://xml.apache.org/xerces2-j/index.html>

## AUTHOR BIO

Frank Neugebauer is a member of the Business Consulting Services division of IBM Global Services specializing in distributed solutions based on IBM's Insurance Applications Architecture (IAA). He has contributed several articles to leading industry publications and served as technical editor for the Pearson Technology Group books *Teach Yourself XML in 21 Days* and *XSL Formatting Objects Developer's Handbook*.

NEUGGS@NETSCAPE.COM

### LISTING 1 • The users.xml sample file

```
<?xml version="1.0" encoding="UTF-8"?>
<Users>
  <User>
    <LastName>Glatz</LastName>
    <FirstName>Richard</FirstName>
    <DOB>03/28/1969</DOB>
  </User>
  <User>
    <LastName>Maher</LastName>
    <FirstName>Kenneth</FirstName>
    <DOB>03/26/1969</DOB>
  </User>
  <User>
    <LastName>Shalabi</LastName>
    <FirstName>Reyad</FirstName>
    <DOB>09/07/1966</DOB>
  </User>
  <User>
    <LastName>Neugebauer</LastName>
    <FirstName>Frank</FirstName>
    <DOB>04/10/1969</DOB>
  </User>
</Users>
```

### LISTING 2 • The getResult() method of the XSLTOutputBean class

```
public String getResult() {
    try {
        // 1. Instantiate a TransformerFactory.
        TransformerFactory tFactory =
            TransformerFactory.newInstance();

        // 2. Use the TransformerFactory to process the
        // stylesheet and generate a Transformer.
        Transformer transformer =
            tFactory.newTransformer(
                new StreamSource(
                    new URL(theXSL).openStream()));

        // 3. Use the Transformer to transform an XML Source
        // and
        // send the output to a Result object.
        StringWriter theWriter = new StringWriter();
        transformer.transform(
            new StreamSource(new URL(theXML).openStream()),
            new StreamResult(theWriter));
    }
}
```

```
theResult = theWriter.toString();
} catch (Exception e) {
    e.printStackTrace();
}

return theResult;
}
```

### LISTING 3 • "Users" template

```
<xsl:template match="Users">
  <TABLE cellpadding="5">
    <TBODY>
      <TR>
        <TH>Last Name</TH>
        <TH>First Name</TH>
        <TH>DOB</TH>
      </TR>
      <xsl:apply-templates select="User">
        <xsl:sort select="./LastName"/>
      </xsl:apply-templates>
    </TBODY>
  </TABLE>
</xsl:template>
```

### LISTING 4 • The Action and JSP mappings within struts-config.xml

```
<!-- Global Forwards -->
<global-forwards>
  <forward name="output"
    path="/userOutput"
    className="org.apache.struts.action.
      ForwardingActionForward">
  </forward>
  <forward name="failure"
    path="/error"
    className="org.apache.struts.action.
      RedirectingActionForward">
  </forward>
</global-forwards>

<!-- Action Mappings -->
<action-mappings>
  <action name="inputPageForm"
    path="/inputAction"
    scope="session"
    type="com.neuggsville.actions.
      InputActionAction">
  </action>
</action-mappings>
```

Download the Code  
[www.sys-con.com/xml](http://www.sys-con.com/xml)

# SaturdaySessions

Alan Williamson  
JDJ Editor-in-Chief

We understand the pressures of work and how difficult it can be to get time off. That is why we have designed this workshop to be held in one day and, as a special bonus, on the weekend, so no days off from work. **Your boss will be happy!**

## JDJ Workshop with Alan Williamson

M	T	W	T	F	S	S
---	---	---	---	---	---	---

Coming to you...

**April:**

NEW YORK  
WASHINGTON, DC

**May:**

BOSTON  
TORONTO

**June:**

ATLANTA  
RALEIGH

# Performance > Efficiency > Reliability

**This one-day intensive workshop is designed for developers who wish to increase the efficiency and reliability of their code development.**

*What you will receive...*

- ✓ INTENSIVE ONE-DAY SESSION
- ✓ DETAILED COURSE NOTES AND EXCLUSIVE ONLINE RESOURCES
- ✓ JDJ CD ARCHIVE

- 1) The day will begin by looking at the various hints and tips you can utilize at the code level to improve the quality and reduce the number of bugs you have to contend with.
- 2) The next part will look at Apache's Ant and how you can use this freely available tool for your own development, irrespective of your IDE.
- 3) Last, and most important, as the old saying goes: "You can never do enough testing." This session will look at JUnit and show you how to start building test harnesses for your code so you can begin your testing strategy.

### >Performance

Java is a powerful language. While it offers a rich array of tools, the fundamentals mustn't be overlooked. Improving your code at the core layer will result in great improvements in efficiency and produce (hopefully) less bugs. We'll look at the do's and don'ts of programming and learn many hints and tips that will accelerate your Java coding.

### >Efficiency with Ant

Apache's Ant is a powerful scripting tool that enables developers to define and execute routine software development tasks using the simplicity and extensibility of XML. Ant provides a comprehensive mechanism for managing software development projects, including compilation, deployment, testing, and execution. In addition, it is compatible with any IDE or operating system.

### > Reliability with JUnit

A critical measure of the success of software is whether or not it executes properly. Equally important, however, is whether that software does what it was intended to do. JUnit is an open-source testing framework that provides a simple way for developers to define how their software should work. JUnit then provides test runners that process your intentions and verify that your code performs as intended. The result is software that not only works, but works in the correct way.



**To Register**

**[www.sys-con.com/education](http://www.sys-con.com/education)**

**Call 201 802-3058**

SPONSORED BY

**JAVA DEVELOPER'S JOURNAL**

PRODUCED BY  
**SYS-CON EVENTS**

PRESENTED BY  
**i-TECHNOLOGY EDUCATION**



WRITTEN BY KEVIN HUCK

# Leveraging Value of Disparate Information Assets

## XML to the rescue

**B**usinesses today are trying to cope with an overabundance of electronic information. Beyond the silos of structured data that have been managed with traditional database technologies is a need to manage the ever-increasing deluge of unstructured data types and sources. E-mails, spreadsheets, Office documents, HTML files, and so on, contain a wealth of information, but are not easily accessible across the organization.

Content management solutions partially address the unstructured information, but cannot necessarily provide management of traditionally defined structured information. Portal technologies help address the delivery of information to the end user, but once again are incomplete storage and management solutions. What is needed is one single mechanism to access, manage, manipulate, and analyze all forms of information, regardless of source or format.

### Traditional Solutions *Virtual database: managing data in silos*

Virtual databases or data access solutions have been around for years. These systems were designed to access silos of managed data, primarily relational and mainframe systems. In their latest versions, they attempt to address unstructured and unmanaged information as well. The goal is to provide a common mechanism to access all corporate information assets. However, there are a number of drawbacks to this approach.

Reaching into a multitude of data sources distributed across an organization generally incurs significant per-

formance problems, particularly when accessing nonrelational sources. Relational database technology can often handle the load placed on extending the user community, but may require significant software and hardware upgrades. However, when real-time accesses reach legacy mainframes designed for batch transactions, upgrades often aren't feasible. The problem worsens when accessing unmanaged sources like documents, spreadsheets, or HTML and flat files. These systems may be offline during a given query, or are nontransactional and cannot deal with multiple simultaneous requests or updates, and leave significant security holes. There is no easy way to track users who make changes to files. Furthermore, an unknowing user directly accessing the files may inadvertently change the contents and/or format and "break" the mappings associated with the data source.

Additionally, this approach is generally view-only and does not allow users to update the information; they must go to the original source or application to make changes - leaving the solution incomplete.

Finally, performance limitations make it unfeasible to perform any interactive or automated analysis across these disparate sources and data types.

### *EAI: complex and considerable customization*

EAI systems are generally designed to automate application-to-application communication, transactions, and workflow processes. While vital in building automated and collaborative systems, they are generally complex and require considerable custom coding and mapping. EAI systems aren't really

designed to provide a common information model to be reused and consumed by multiple end users. Instead, these systems focus on system-to-system integration. These solutions also generally work only with managed data sets in a transactional manner rather than unstructured/unmanaged documents and files.

### *Data warehouses and data marts: costly and intricate*

Data warehouses and data marts provide excellent mechanisms for high-speed complex analysis by specialized business users. However, these systems achieve their goals by creating complex, highly structured and optimized constructs that are difficult and expensive to create, maintain, and synchronize with back-end sources. Significant analysis and knowledge of what questions are to be asked of the solution must be determined during requirements gathering. Furthermore, these systems don't work well with unstructured or unmanaged sources.

### *Content management systems: limited in granularity*

These systems attempt to aggregate and manage unstructured information such as HTML pages, documents, spreadsheets, audio/video files, and so on. However, they cannot incorporate managed sources such as relational and mainframe databases. Furthermore, they don't allow fine-grained, element-level access or access control; the granularity of access is at the document level. While these systems can work well for unstructured data types, businesses need to blend the two worlds, structured and unstructured, into a common unified solution. To the business, informa-

## AUTHOR BIO

Kevin Huck, chief architect, NeoCore Inc., has been involved in software engineering for nearly 20 years. Most recently, he has been involved with the development of a high-performance native XML database that extends the J2EE platform by offering XML and object-persistence mechanisms. A speaker at numerous small user groups, engineering organizations, and teams on system architecture, XML, and J2EE technologies, Kevin has a bachelor's degree in business administration from Baker University and an MBA in entrepreneurship from the University of Missouri-Kansas City.



tion is information; its form and storage location should not limit the ability to access, update, manipulate, or analyze the information.

Each of these solutions addresses a need within corporate IT departments. However, none of them were developed to address the aggregation of disparate data sources of all types and formats. A new approach, one based on the "lingua franca" of information, XML, is needed, and unique properties and characteristics are required to fully address the issue.

## XML to the Rescue

XML has taken the information technology industry by storm. While XML was originally adopted as a mechanism for information interchange, companies are seeing the value of storing and managing information in XML form. The advantages are many. First, XML is the first standards-based means of describing "information" – data with context, making XML self-describing. A human can read and understand an XML document without technical knowledge or knowledge of any underlying systems. XML is extensible, allowing document types to easily be changed and adapted as business needs change. XML also supports flexible data types and does not impose type or length restrictions. Finally, XML supports heterogeneity, the ability to support hundreds, even thousands, of document types, which is crucial to managing previous unstructured information such as Office documents, HTML files, e-mail, and so on.

### Easy to understand

The basics of XML are easily understood by most business users, and there are now a host of third-party tools that make it easy to create XML documents and schemas and to extract/transform/load (ETL) data to and from non-XML sources such as relational databases, mainframes, spreadsheets, flat files, etc. Common desktop applications such as Microsoft Office, Web browsers, editors, and development environments now have varying support for XML, quickly making XML as ubiquitous as HTML became in the '90s.

Creating and migrating corporate information assets into XML is only one step. To fully leverage XML's capabilities, an appropriate XML database is required, allowing the strengths of each of the aforementioned approaches to be combined into one flexible, adaptive system without most of the complications and penalties. The optimal system

is entirely self-constructing (can accept any XML document without database design or indexing instructions) so that it can assimilate all types of information, structured and unstructured, and provide a high-speed, common-access mechanism based on current and emerging Web standards such as SOAP, HTTP, and XPath/XQuery. It is also fully transactional, so it can support updates as well, and feed these updates synchronously or asynchronously to the original source. It should serve as the database of record for new applications that are XML-centric. This allows a business to turn all digital assets into a set of services available throughout the organization, regardless of original source or format.

### Additional benefits: efficiency and cost effectiveness

Furthermore, since a self-constructing XML database can support hundreds, even thousands, of different data schemas without physical database design, many original sources can be eliminated. For example, if product and pricing information is currently kept in a series of spreadsheets shared on a disk drive, a simple, inexpensive, and cost-effective application can let users view and manage this information directly in an XML database, and the original spreadsheets are no longer necessary. Generally, this would be a fairly complex IT effort, particularly as the number of spreadsheets increases. However, these simple data management systems can be built using an extreme rapid application development (XRAD) methodology, leveraging today's powerful Web tools like .NET, ColdFusion, JBuilder, and so on. Tasks that once took months can now be accomplished in days, and the information can now be shared, managed, tracked, and secured much more efficiently and cost effectively.

Finally, since the XML database acts as a high-speed cache, and is designed as a piece of corporate intranet/Internet architecture, it acts as a "shock-absorber" against the real-time demands placed on original data sources.

### Not just information management

Going far beyond the benefits of providing a common mechanism to manage all types of information, regardless of source or format, is the ability to view, drill into, and analyze these valuable corporate assets. Since a self-constructing XML database automatically indexes all XML information, very high-speed ad

hoc queries and analysis can be performed.

### Let the business user navigate the information


Generally, business users don't have sufficient technical knowledge to access data in data warehouses and data marts, because the actual implementation does not match the business user's logical view of the information. Mapping business data, often hierarchical in nature, to a series of normalized relational tables results in the absence of context; the user cannot understand what information resides where. How-

**"XML supports heterogeneity, the ability to support hundreds, even thousands, of document types"**

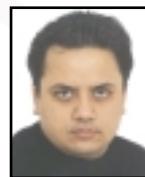
ever, users can create logical XML information models where the context, or tags, is informative, and the relationships, via tag hierarchy, are obvious and well understood (such as in file systems), and which closely maps to the original business requirements. They can then easily understand and navigate the information without detailed technical knowledge. This reduces the need for specialists and limits demands on overburdened IT staff.

Because all queries are index driven, no additional work or tuning is needed when a business user determines another question for the system. Simply using an interactive analysis tool built on top of the XML database, or by crafting XPath/XQuery statements, new answers to previously unconsidered questions are delivered immediately.

### IT Shouldn't Mean High Costs

In today's cost-conscious corporate environment, IT departments face an ever-increasing challenge to do more with less. Trying to get a handle on a growing set of data sources and formats with fewer resources requires a different approach. A self-constructing XML database is designed to address this specific problem while leveraging widely adopted Internet standards and protocols, allowing companies to reduce cost, development time, and complexity, while increasing the value of their information assets. 

 KHUCK@NEOCORE.COM



Hitesh Seth, editor-in-chief of XML-Journal and XML Track chair for Web Services Edge, is the chief technology officer of ikigo, Inc., a provider of business activity monitoring solutions.

REVIEWED BY HITESH SETH

# XMLBeans

## The Best of Both Worlds

**J**ava and XML – portable code and portable data.” Even though this saying has been around since Java developers began using XML, developers have always faced a general XML programming-related productivity problem: manipulating XML content is rather different from manipulating Java objects.

This problem has been amplified with the emergence of XML Schema, the W3C standard type system for XML documents. While XML Schema provides a rich type system for XML documents, it isn't simple. Above all, the XML Schema type system differs significantly from the Java type system (e.g., the notion of simple types versus complex types, a rich set of predefined simple types, schema by restriction, etc.)

Several alternatives have already surfaced to solve some of these problems. For example, we have the tree-based Document Object Model (DOM), and the lower-level, events-based SAX (Simple API for XML), implemented by Apache Xerces, the almost de facto standard Java parser implementation. For Java programmers searching for a more natural object-oriented mechanism, there's the Java API for XML Binding (JAXB), which was recently released in its 1.0 specification. JAXB (covered in *XML-Journal*, Vol. 4, issue 1) intends to create a Java-XML binding framework that provides Java programmers with more natural object-oriented access to the underlying XML data. JAXB also supports XML Schema-based type binding (with some exceptions).

Recently, BEA released a beta of XMLBeans, an XML token stream-based technology that provides easy navigation of XML documents using a combination of interfaces – Java types (which

have been mapped to the relevant XML Schema-based types) and native XML cursor- or XQuery-based access. The result is a technology that combines the benefits and convenience of manipulating XML data using a familiar Java type tree but doesn't sacrifice access to the underlying XML infoset.

### Key Concepts

XMLBeans has three APIs, or mechanisms to manipulate XML data:

1. **XML Cursor:** The XMLBeans implementation provides a simple DOM-like tree navigation API to the underlying XML document. This is available even if XML Schema (xsd files) are not compiled.

**“XMLBeans clearly provides developers a viable, useful, and productive alternative for a number of scenarios”**

2. **JavaBeans:** A set of XML Schema (xsd files), known as an XSD type system, is processed by a schema compiler to generate a set of Java classes and interfaces corresponding to the XML Schemas. A one-to-one relationship is maintained – one XSD type corresponds to a single Java type (or interface). All schema types are derived from the base class `XmlObject`. It's also useful to note that XMLBeans objects are serializable, so they can be transmitted over RMI boundaries. Similar to Java's inherent reflection capabilities, XMLBeans also provides capabilities to introspect the XSD Schema type through a set of classes. In the current beta implementation, the models available for schema compilation are a hosted compiler and the built-in sup-

port within BEA WebLogic Workshop.

3. **XQuery:** With Apache Xerces2, XMLBeans provides the capability to run XQuery-based queries to query XML for specific data.

What differentiates XMLBeans from other Java binding technologies such as JAXB and Castor is the fact that XMLBeans supports two synchronized XML access models (see Figure 1) – one to the underlying XML content as well as one to the strongly typed Java classes. In addition, one of the primary objectives for XMLBeans (in its final release) is to provide 100 percent support of all features and capabilities in XML Schema (as specified by the W3C Recommendation).

### Usage Scenario

Let's take a simple schema that represents a simplified order document. Listing 1 (as well as Figure 2) shows the XML Schema of an order (code for this article is available at [www.sys-con.com/xml/source.cfm](http://www.sys-con.com/xml/source.cfm)). Listing 2 shows an example XML instance for the schema.

The next step is to generate a set of classes that represent the Order XSD type system. To do this, upload the `Order.xsd` file to BEA's hosted XMLBeans compiler on <http://dev2dev.bea.com>, which returns a JAR file (and source, if required) of the resulting interfaces/classes. This JAR file can then be included in any project (EJB, Web, or even a stand-alone Java application) to process XML documents for the particular type. Listing 3 shows one such application – calculating the total order amount.

The JavaBeans-based type bindings generated by XMLBeans and JAXB are quite similar by design, which indicates that XMLBeans can almost be an upgradeable option to existing JAXB developers. It also hints at a possible

convergence (hopefully!) of the two initiatives.

The fact that XMLBeans preserves the underlying XML content can be illustrated by invocation of `xmlText()` on the complete document object or a subelement such as `lines[i]`. You'll find that the call preserves the comments and the order of elements in the original document. This is particularly important in many application scenarios. For instance, if an XML-based configuration file is used, comments will typically be added to enhance readability. XMLBeans can be used to manipulate the content (read/write/update) but will also preserve the important information in it, such as comments. Another scenario in which preserving the original XML text is very important is an XML-based workflow application in which the content is significantly enhanced/processed as it follows the various activities in the application. Checking document validation is simply calling the `validate` method on the `XmlObject` (which is the super class of `OrderDocument`).

So far we've seen a mechanism by which we're simply reading the XML content; the type system generated by XMLBeans also includes full support of modification of XML elements and tags using the generated classes. Listing 4 shows one such example, in which a business rule, "10% discount to all items", is applied to the original XML document.

Running the sample program in Listing 4 will also illustrate that XMLBeans preserves the underlying XML content.

## Support for XQuery

The example in Listing 5 uses XQuery to enlist the various lines in which the quantity is greater than 50. Of course, this is a limited usage of XQuery considering the various features that the new query language supports; however, it illustrates the support in XMLBeans. XQuery support does require Apache Xerces classes to be present in the class-path. (*Note:* for using the schema within Java applications, the only class required other than the generated schema types is the `xmlbeans.jar` runtime.)

## BEA WebLogic Workshop

Support for XMLBeans is a key highlight of the new BEA WebLogic Workshop 8.1. The new version (recently announced at BEA eWorld) supports XMLBeans using a simple drag-and-drop mechanism. Once a schema file is uploaded into a current project, the corresponding schema JAR files are auto-

matically created and the files are added to the project, ready to be available for the “code-complete” features in the tool. This capability is illustrated by Figures 3 and 4.

Note that while BEA WebLogic Workshop provides out-of-the-box support for XMLBeans-based schema type support, any IDE can be used to import the generated class files used in applications.

## Availability

The beta is now available from <http://dev2dev.bea.com>. Currently, XML-Beans is available in two ways: as a hosted service from dev2dev.bea.com, and with WebLogic Workshop 8.1, which is available for download from dev2dev as well. One thing to be mindful of is that XML-Beans requires J2SE v1.4.

According to BEA officials, BEA plans to continue the support for hosted schema compilation service, which compiles schemas into Java classes. BEA is further exploring standardization efforts and ways to make XMLBeans broadly available through channels such as open source. However, there has been no official word on that yet. Support is available for XMLBeans through dev2dev-based newsgroups.

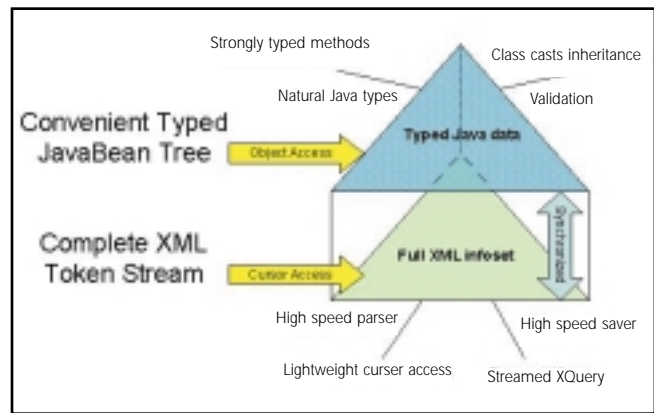
## Conclusion

There has been some discussion in the industry that we need a different programming language with more native support for XML. We've also had discussions about how XML should be supported as a native data type in popular programming languages. XMLBeans clearly provides developers a viable, useful, and productive alternative for a number of scenarios in which they need to work with complex XML Schemas and have access to the underlying XML infoset. I'd really like to see technologies like XMLBeans merge and improve existing industry standardization initiatives such as JAXB (out of the Java Community Process).

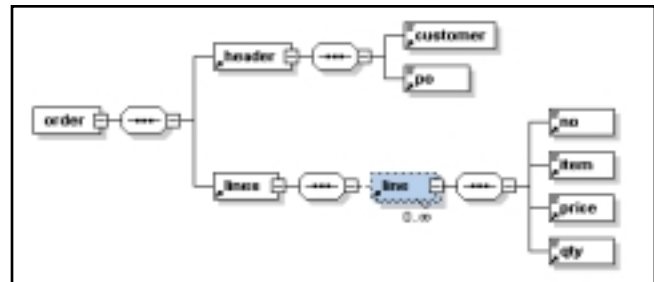
## Resources

- *Hosted XMLBeans Service (Beta):*  
<http://dev2dev.bea.com/technologies/xmlbeans/index.jsp>
- *WebLogic Workshop 8.1 Beta:*  
<http://commerce.bea.com/showproduct.jsp?family=WLW&major=8.1&minor=-1>
- *XMLBeans documentation:*  
<http://workshop.bea.com/xmlbeans/docindex.html>

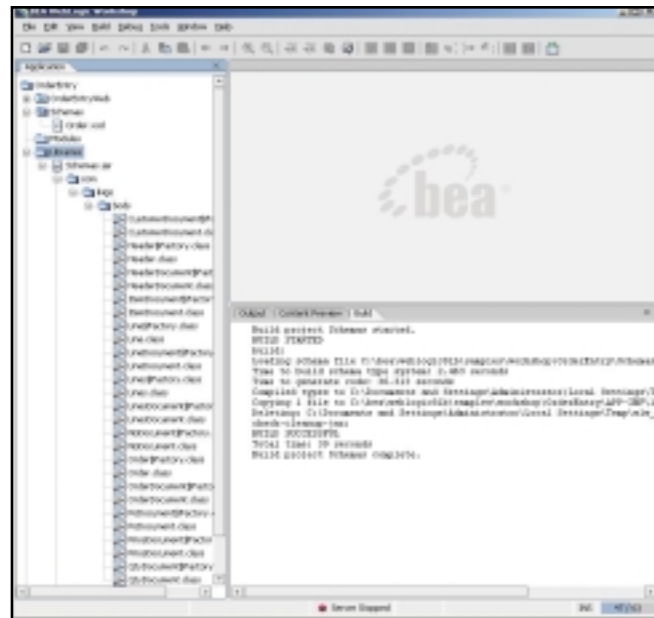
**HITESH@SYS-CON.COM**



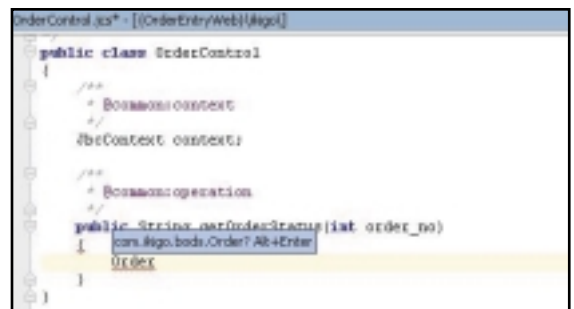
**Figure 1 • XMLBeans architecture**



**Figure 2 •** Visual representation of order schema (using XMLSpy)



**Figure 3 • XMLBeans support in BEA WebLogic Workshop 8.1**



**Figure 4 • Code complete feature**



Hynes Convention Center: Boston, MA

# XML Edge 2003 East

INTERNATIONAL XML CONFERENCE &amp; EXPO



*When SYS-CON Media's sister company, SYS-CON Events, began preparing last year for this spring's "XML Edge" Conference & Expo, one consideration was paramount: every effort in the nine-month preparation cycle should be geared toward making it indisputably the world's largest independent Java, .NET, XML, and Web services event.*

That particular mission was accomplished on March 18-20, 2003, at the centrally located Hynes Convention Center in Boston, Massachusetts, when Web Services Edge 2003 East made its mark right from the get-go, with delegates from a wide variety of companies both technologically and geographically. Not only had they been attracted by the specific session tracks for Java, .NET, XML, and Web services, they had also come to take advantage of the all-day *i*-technology tutorials, whether it was the Sun Microsystems Java University, the IBM XML Certified Developer Fast Path, Russ Fustino's .NET workshop (Russ' Tool Shed), or Derek Ferguson's Mobile .NET tutorial.

The show opened with a very well-

attended keynote from Oracle's John Magee, VP of Oracle9i Application Server. Magee stressed that the key to understanding why Web services, unlike its distributed-computing forerunners like COM and CORBA, is prevailing in the enterprise space is that Web services do more than merely enable interoperability between platforms and integration between applications – they also do so simply.

What drives their simplicity, Magee explained to the audience, is standards.

The afternoon keynote offerings on Day One of the conference were equally well received. First came a panel coordinated by the Web Services Interoperability Organization (WS-I). The WS-I is an open industry organization chartered to promote Web

services interoperability across platforms, operating systems, and programming languages, and the panel discussion took place against the backdrop of the WS-I Basic Profile 1.0, consisting of a set of nonproprietary Web services specifications. The working draft for this, the audience learned, was approved just four weeks before the conference.

But security, the panel agreed, was the primary priority. Now that corporations like Merrill Lynch and DaimlerChrysler have joined the organization, ensuring that everyone adheres to the same specification is more important than ever. Web services is moving beyond mere SOAP, WSDL, and UDDI toward addressing security, messaging, reliability, and transactions. Eric Newcomer, chief technology officer of IONA Technologies, emphasized the importance of the World Wide Web Consortium (W3C) approach to these challenges, an effort that centers on the W3C's Web Services Specification Effort.

The Web services keynote panel was quickly followed by the highlight of Day One for many of the delegates gathered in





the keynote hall: an address by Miguel de Icaza, the impossibly young and extremely gifted founder and leader of the GNOME Foundation, cofounder of Ximian, Inc., and .NET expert extraordinaire – as anyone needs to be who leads a project designed to port .NET to the Linux operating system.

The Mono Project, as de Icaza's project is called, clearly fascinated the broad mix of developers attending the conference.

After explaining that GNOME – a desktop development platform and suite of productivity applications – is his company's key focus and is mostly developed in C, C++, Python, and Perl, he went on to recount how for every new GNOME API (GNOME is component-oriented and supports many programming languages), GNOME developers needed to develop language-specific bindings. Thus .NET, which also addresses the multilanguage problem, was of immediate interest to de Icaza.

As soon as he learned about the .NET Framework, he told the spellbound audience, he got excited – a single Virtual Execution System for multiple languages, with a large and reusable factored class library, that was, in his view, just what was needed. As well as being a new way to do things, .NET's rich support for interop (COM, P/Invoke) meant you didn't have to rewrite everything all at once.

And so Mono was born: an open-source .NET Framework implementation.

It's based around the CLI ISO standard, de Icaza continued. It has a CLI-compliant execution system and a x86 JIT compiler. It's supported by Windows, BSD, Linux, and Solaris, and there has been lots of progress on the class libraries.

The Windows support, de Icaza said, was merely a function of the fact that 60% or so of Mono developers have a Windows background. Some of the code contributed to Mono was funded by Microsoft grants, he added.

At the end of his keynote address, scores of developers of every stripe got up from their chairs and surrounded de Icaza for further questions. The response to his good humor, rapid delivery, technical savvy, and



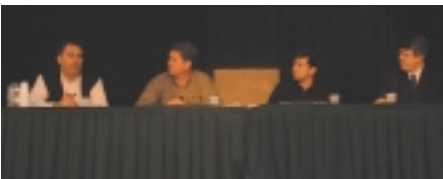
**John Magee, Oracle:**  
"Developing in a  
Services-Based World"



**Mark Herring,**  
**Sun Microsystems:**  
"Bridging the Gap Between  
WS-Myth and WS-Reality"



**Miguel de Icaza,**  
**Ximian:**  
"The Mono Project"



**WS-I Panel Discussion:**  
"A Road Map for Web Services Standards"



**Java Panel Discussion:**  
"The Future of Java"

sheer charm had been overwhelming and with his keynote, Web Services Edge 2003 (East) passed a significant milestone: no previous conference in the series had ever included so wide a range of technical content.

Day Two saw Sun's Mark Herring take the keynote stage and his mastery of the whole Web services paradigm was clearly in evidence. Extended coverage of both his Java keynote and the subsequent keynote address by Jesse Liberty are available on the main conference Web site, [www.sys-con.com/WebServicesEdge2003East](http://www.sys-con.com/WebServicesEdge2003East).

The closing keynote discussion panel, which for many turned out to be the high point of the entire keynote program, was a wide-ranging and a sometimes heated debate about "The Future of Java." The whole intense and highly interactive hour exemplified very well how a SYS-CON *i*-technology conference program differs from that offered by any other conference organizer. This was panel discussion at its best.

True to the enormously close links that **Java Developer's Journal** enjoys with the software development industry, the participants in this final panel at Web Services Edge 2003 (East) had come to Boston

from far and wide. Sun's chief technology evangelist Simon Phipps had flown over from the UK and BEA's director of technology evangelism Tyler Jewell had traveled from Los Angeles. Sonic Software's VP and chief technology evangelist Dave Chappell may have nipped across from Bedford, MA, but Aligo's CTO Jeff Capone had flown in from San Francisco, and JBoss founder Marc Fleury had come up from the JBoss Group's company's HQ in Atlanta, Georgia.

We fully expect the next Conference & Expo, Web Services Edge (West) in October, to be equally chock-full of the movers and shakers of the software development industry as it continues its headlong progress toward distributed computing with full application integration and interoperability.

All in all it was a marvelous conference, and the Expo hall was intensely busy from the moment it opened to the moment it closed two days later.

This is not the end of the Web services "story," nor is it even the beginning of the end; but March 18–20 in Boston's Hynes Convention Center may well have marked the end of the beginning.

Come join us for Phase Two...in October in California. ☘





WRITTEN BY RAFAH HOSN, GERALD MCCOBB, T.V. RAMAN

# Versatile Multimodal Solutions

## The anatomy of user interaction

### AUTHOR BIOS

T.V. Raman is an accomplished computer scientist with more than eight years of industry experience in advanced technology development. He has authored 2 books and filed over 20 patents. T.V. participates in numerous W3C working groups and authored Aural CSS (ACSS); in 1996 he wrote the first ACSS implementation.

Gerald McCobb is an Advisory Engineer with IBM's Pervasive Computing division. He is currently working on multimodal development related to embedding speech technology on small devices. He is also the secondary IBM representative to the W3C Multimodal Working Group.

Rafah A. Hosn is a senior software engineer with IBM research. She is a member of the Interaction Middleware and Standards research group and is currently working on the design and development of WebSphere middleware to support multimodal interaction.

**U**ser interaction is about creating an effective man-machine conversation that leads to rapid task completion. With this in mind, we can factor the typical application into the data model that holds the current interaction state, user interface components that render this state, and interaction behavior that is determined by the active event handlers.

Today, Web interaction is authored in markup languages like XHTML, with the host document providing the data model and user interface components, and the host browser implementing the DOM2 eventing loop for bringing the Web interaction to life. The W3C architecture exemplifies this separation in its current developments:

- Widespread adoption of CSS separates content from presentation.
- W3C XForms separates the data model from the user interaction.
- W3C XML Events exposes the DOM2 Events interface to the XML author, thereby making the original vision of *the document is the interface* a reality.

Notice that in this architecture, the richness of end-user interaction is a function of the available user interface events and event handlers that are available to respond to a given event. As the next evolution in user interfaces, multimodal interaction can be integrated into this evolving framework by enabling the Web author to attach rich voice handlers that implement spoken dialogs to aid in rapid task completion. A means to achieve this end was first detailed in XHTML+Voice (X+V). This article describes X+V 1.1, an update to X+V that integrates the results of more

than two years of experience gained by implementing multimodal solutions using this framework.

The remaining sections of this article summarize the additions to X+V and illustrate their use in creating multimodal interaction that leverages mixed-initiative VoiceXML dialogs. Formal descriptions of these additions can be found in the X+V 1.1 specification; here we'll focus on motivating these additions and explaining their use.

### Aural CSS – Speaking in Style

Aural CSS (ACSS) – part of W3C Cascaded Style Sheets (CSS) – enables the XHTML author to specify style rules for aural presentation. X+V 1.1 leverages Aural CSS by allowing the XHTML author to attach an aural style rule to a CSS class. In the following example, we illustrate a stylesheet fragment that attaches an aural style to XHTML `p` elements having class `romeo` and `juliet`.

```
P.romeo { voice-family: male;
           volume:
           loud; pause-before: 20ms; }
P.juliet { voice-family: female;
           volume: soft; }
```

Create the content in style. Identify `p` elements in the XHTML document using class `romeo` or `juliet` as shown below.

```
<body ev:event="load"
      ev:handler="sayHello">
  <p id="hello_romeo" class="juliet">
    Romeo, Romeo, where art thou?
  </p>
  <p id="hello_juliet" class="romeo">
    I am here.
  </p>
</body>
```

Finally, define the voice handler invoked in the above fragment to speak the contents of the `p` elements. Contents from the XHTML document are accessed from within the `vxml:prompt` element using new X+V attribute `xv:src`. Using attribute `src` in this manner is consistent with the forthcoming W3C XHTML 2.0, which uses this attribute to allow authors to specify the contents of elements indirectly; X+V 1.1 extends element `vxml:prompt` with attribute `xv:src` to enable equivalent functionality when authoring multimodal interaction.

```
<vxml:form id="sayHello">
  <vxml:block><prompt
xv:src="#hello_romeo"/>
    <prompt
xv:src="#hello_juliet"/>
  </vxml:block>
</vxml:form>
```

See Listing 1 for the complete example.

### Two-Way Synchronization – One Handler to Bind Them

In a multimodal interaction, user input obtained via a given interaction modality needs to be made available to all participating interaction modalities. In practice, this reduces to two-way synchronization between the visual and auditory modalities when authoring multimodal interaction using X+V and the HTML forms module. Note that the transition to W3C XForms will enable the synchronization of more than two modalities since XForms provides an explicit data model that records interaction state. When we created X+V 1.0, we wanted to provide a smooth transition for today's Web authors using HTML forms; to ease this transition, we

enabled implicit two-way synchronization between the visual and spoken interaction states. Experience in authoring applications using such implicit synchronization showed that there is value to exposing this two-way synchronization to the XML author; to this end, X+V 1.1 defines declarative construct `sync`, which can be used to associate components of the visual and aural interaction state. The declarative nature of element `sync` is particularly important when deploying X+V solutions to thin clients that may not be able to support a full scripting environment.

Element `sync` can be thought of as a declarative event handler that synchronizes the two interaction state components being linked. We describe the use of element `sync` in the remainder of this section, with illustrative code fragments taken from the complete example shown in Listing 2.

Element `sync` uses attributes `input` and `field` to specify the two interaction state components to be synchronized. The attribute names were chosen to match the interaction components most commonly used in the visual and aural modalities, respectively. Value of attribute `input` holds the value of attribute `name` from the HTML form control being synchronized. Notice that in the absence of a data model in HTML forms, this `name` attribute also names the location where the control stores user input. Attribute `field` holds the id of the VoiceXML `field` to be synchronized.

In the mixed-initiative example shown in Listing 2, we've declared that the visual input controls that collect the *hotel* and *city* should be synchronized with the corresponding VoiceXML fields via the following statements:

```
<xv:sync input="city"
field="#field_city"/>
<xv:sync input="hotel"
field="#field_hotel"/>
```

Notice that the value of attribute `field` is a URI, i.e., as in the rest of X+V, the VoiceXML elements may appear either within the XHTML document, or in a separate XML file. Using this addition to X+V, the author can enable several interaction metaphors that will be described in the remainder of this section.

### Spoken Input with Visual Confirmation

The mixed-initiative VoiceXML dialog shown in Listing 2 is activated when the XHTML form gets focus. The

mixed-initiative dialog collects the hotel and city names when the user speaks:

"I would like to stay at the Chicago Airport Hilton"

and the values are synchronized with the visual interaction components. The user gets immediate visual confirmation as a result of the values being synchronized.

For the same utterance as above, the voice browser recognizes the hotel, but fails to identify the city. Tapered prompts within the VoiceXML handler lead the user through the rest of the task. Synchronizing the fields helps the user get immediate feedback as to the portion of the task that remains to be completed.

### Talk and Type

Notice that resorting to tapered prompts when the mixed-initiative dialog fails to get all the desired values turns the mixed-initiative dialog into a directed dialog. When using multimodal interaction, mixed-initiative dialogs can also turn into a directed dialog if the user uses the keyboard or stylus after having provided initial speech input. To continue the example, after speaking the previously mentioned utterance, the user might explicitly move the focus to one of the visual input controls using the keypad or pointing device. Such user action can be thought of as an implicit *escape* from the mixed-initiative dialog. The X+V execution model specifies that only one voice handler is active at a given time; we leverage this fact in the example shown in Listing 2 by attaching simple VoiceXML-directed dialogs to each of the visual input controls for event focus. As a result, if the mixed-initiative dialog fails to perform satisfactorily for a given user, that user can provide input via the keypad or pointing device to transition to a directed dialog. To complete the example, tabbing to the *hotel* input control would result in the directed dialog for that field being activated, and this would first cancel the mixed-initiative dialog.

### Talk or Type

In certain situations, the author might wish to create a multimodal experience in which the user is initially prompted with a voice prompt, along with the activation of a mixed-initiative dialog. If the user starts providing input

via the keypad, the author might wish to keep the mixed-initiative dialog active, rather than resorting to a directed dialog where the user hears a spoken prompt for each field. This form of interaction can be enabled by simply dropping the binding of the directed dialog handlers to the individual visual controls.

### Cancel – Speech Is Silver, but Silence Is Golden

We described the implicit canceling of a running VoiceXML dialog in the previous section. This *cancel* action is made available to the XHTML author via handler `cancel`. We use this in Listing 2 where we attach handler `cancel` to XHTML control `reset`. The effect is that if the user resets the form via the non-speech modality, the currently active voice handler is canceled, and the user interaction returns to its initial state.

### Deploying X+V Multimodal Solutions

Reuse of content assets is one of X+V's strongest features. X+V enables speech interaction specialists to create high-quality spoken dialogs that can be easily plugged into well-designed visual interfaces. In this context, it is worth noting that multimodal solutions need to be deployed in a variety of device and network environments in contrast to traditional desktop Web interfaces that adhere to the *one browser fits all* model. What follows is an overview of the various deployments enabled by the X+V architecture.

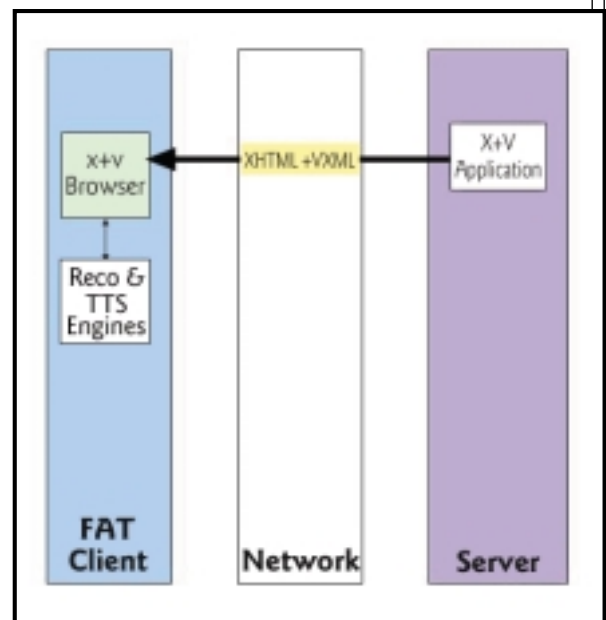


Figure 1 • Fat-client architecture



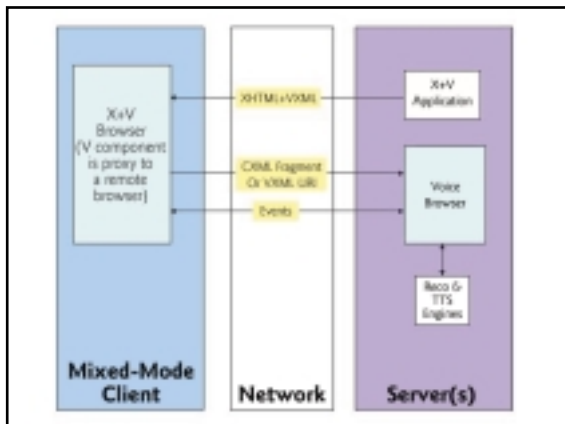


Figure 2 • Mixed-mode client

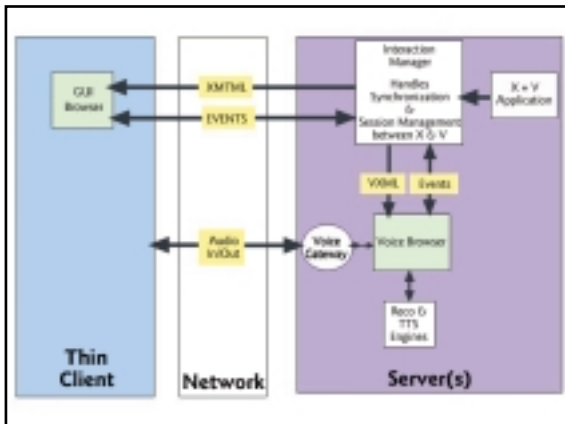


Figure 3 • Thin-client architecture

### PDA

A PDA capable of local speech processing can execute the voice handlers shown in Listing 2 locally on the device. Visual and spoken modalities can be tightly synchronized to provide a rich multimodal user experience. Figure 1 shows a fat-client architecture in which the multiple modalities are processed locally on the client.

### Mixed-mode client

A PDA or smart phone may not always have sufficient resources to perform every speech processing task. In

this case, complex speech processing, and consequently, the execution of the associated voice handlers can be off-loaded to a voice server running on the network. Notice that the X+V design is particularly well-suited to this kind of deployment since the voice handlers do not have to reside in the same document as the rest of the XHTML markup. In Figure 2, an overview of this architecture, the mixed-mode client off-loads some processing to a network server. The X+V markup can be distributed as needed.

This feature is a consequence of reusing XML Events within X+V for creating event bindings. XML Events has been designed to operate in the hypertext environment of the Web, and a key design feature is that event handlers can be referred to via a URI. We emphasize this point in this article because the X+V specification itself does not highlight this fact, since we got the feature for free by virtue of reusing XML Events. Since the publication of X+V 1.0, authoring of remote voice handlers has been one of the most frequently asked questions from X+V partners, and we highlight the solution here for this reason.

### Thin client

Consider a cellphone that has sufficient speech processing capability to implement only simple command and control navigation. All other speech functions are performed by off-loading the processing to a network-based voice server. In such an environment, the X+V document illustrated in Listing 2 can be factored into separate XML documents that hold the visual XHTML markup and event bindings, and a separate XML document that holds the voice handlers. The event bindings in the XHTML markup can be updated to point to the network location where the XML document containing the voice handlers will be processed. Figure 3 shows an overview of this architecture. A distrib-

uted multimodal client can carry out processing of user input at different points on the network. The markup solution provided by X+V allows for the relevant markup to be transmitted and cached where it will eventually be used. This can save valuable bandwidth and XML processing on thin clients.

This has the advantage that only the visual markup is transmitted to the thin client, i.e., the deployment respects the separation of concerns inherent in this deployment by not transmitting the voice markup to the thin client. This can save valuable bandwidth and processing cycles on the thin client. The voice handlers are available and preloaded into the voice server which can consequently be extremely responsive when it receives a request for invoking one of the specified voice handlers from the thin client. XML Events bridges the two execution environments, i.e., the visual browser on the thin client and the voice browser on the network.

Finally, notice that the completely declarative nature of the X+V document is a major win in reliably deploying the multimodal application to devices such as cellphones which typically lack support for a full scripting environment. ☎

### References

- *XML Events*: [www.w3.org/TR/xml-events](http://www.w3.org/TR/xml-events)
- *CSS*: [www.w3.org/TR/CSS2](http://www.w3.org/TR/CSS2)
- *W3C XForms*: [www.w3.org/TR/xforms](http://www.w3.org/TR/xforms)
- *XHTML + Voice*: [www.w3.org/TR/xhtml+voice](http://www.w3.org/TR/xhtml+voice)
- *X + V 1.1*: [www-3.ibm.com/software/pervasive/multimodal](http://www-3.ibm.com/software/pervasive/multimodal)
- *W3C XHTML 2.0*: [www.w3.org/TR/xhtml2](http://www.w3.org/TR/xhtml2)

RHOSN@US.IBM.COM

MCCOBB@US.IBM.COM

TVRAMAN@ALMADEN.IBM.COM

### LISTING 1 • Speaking In Style

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:vxml="http://www.w3.org/2001/vxml"
xmlns:ev="http://www.w3.org/2001/xml-events"
xmlns:xv="http://www.voicexml.org/2002/xhtml+voice"
>
  <head>
    <style type="text/css">
      P.romeo { voice-family: male; volume: loud; pause-before:
        20ms; }
      P.juliet { voice-family: female; volume: soft; }
    </style>
    <vxml:form id="sayHello">
```

```
<vxml:block><prompt xv:src="#hello_romeo"/>
  <prompt xv:src="#hello_juliet"/>
</vxml:block>
</vxml:form>
</head>
<body ev:event="load" ev:handler="#sayHello">
  <p id="hello_romeo" class="juliet">
    Romeo, Romeo, where art thou?
  </p>
  <p id="hello_juliet" class="romeo">
    I am here.
  </p>
</body>
</html>
```



## LISTING 2 • Multimodal Interaction Using Mixed-initiative Dialog

```
<?xml version="1.0"?>
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:vxml="http://www.w3.org/2001/vxml"
xmlns:ev="http://www.w3.org/2001/xml-events"
xmlns:xv="http://www.voicexml.org/2002/xhtml+voice"
>
  <head>
    <title>Mixed Initiative Conversational
      Interface</title>
    <!-- first declare the voice handlers. -->
    <!-- VXML form supporting a mixed-initiative
      grammar -->
    <vxml:form id="voice_city_hotel">
      <vxml:grammar src="city_hotel.srgs" type=
        "application/srgs"/>

      <!-- Mixed initiative form begins with initial
        prompt -->
      <vxml:initial name="start">
        <vxml:prompt xv:src="#please_choose"/>
        <vxml:help>
          Please say the name of a city and a hotel
          to make
          a reservation.
        </vxml:help>
        <!-- If user is silent, reprompt once, then try
          directed prompts. -->
        <vxml:noinput count="1"><vxml:reprompt/>
        </vxml:noinput>
        <vxml:noinput count="2">
          <vxml:reprompt/>
          <vxml:assign name="start" expr="true"/>
        </vxml:noinput>
      </vxml:initial>

      <vxml:field xv:id="field_city" name="field_city">
        <vxml:grammar src="city.srgs"
type="application/srgs"/>
        <vxml:prompt>Please choose a city.</vxml:prompt>
        <vxml:catch event="help nomatch noinput">
          For example, say Chicago.
        </vxml:catch>
      </vxml:field>

      <vxml:field xv:id="field_hotel" name="field_hotel">
        <vxml:grammar src="hotel.srgs" type=
          "application/srgs"/>
        <vxml:prompt>Select your hotel.</vxml:prompt>
        <vxml:catch event="help nomatch noinput">
          For example say Hilton.
        </vxml:catch>
        <vxml:filled>
          <vxml:prompt>
            You selected <vxml:value expr="field_hotel"/>.
          </vxml:prompt>
        </vxml:filled>
      </vxml:field>

      <!-- done mixed-initiative voice handlers -->
    <!-- fall back directed dialog handlers -->
    <vxml:form id="voice_city">
      <vxml:field xv:id="field_city" name="field_city">
        <vxml:grammar src="city.srgs"
```

```
type="application/srgs"/>
        <vxml:prompt xv:src="#city_label"/>
        <vxml:catch event="help nomatch noinput">
          For example, say Chicago.
        </vxml:catch>
      </vxml:field>
    </vxml:form>

    <vxml:form id="voice_hotel">
      <vxml:field xv:id="field_hotel" name="field_hotel">
        <vxml:grammar src="hotel.srgs" type=
          "application/srgs"/>
        <vxml:prompt xv:src="#hotel_label"/>
        <vxml:catch event="help nomatch noinput">
          For example, say Hilton.
        </vxml:catch>
        <vxml:filled>
          <vxml:prompt>
            You selected <vxml:value expr="field_hotel"/>.
          </vxml:prompt>
          </vxml:filled>
        </vxml:field>
      </vxml:form>

      <!-- declare inputs synchronized with VoiceXML
        fields -->
      <xv:sync input="city" field="#field_city"/>
      <xv:sync input="hotel" field="#field_hotel"/>
      <xv:cancel id="voice_cancel"
handler="#voice_city_hotel"/>
    </head>
    <body>
      <h1>Mixed Initiative Conversational Interface</h1>

      <p>In this example we demonstrate a mixed-initiative
        dialog. By activating a grammar capable of
        recognizing both cities and hotel names for the
        entire application, the user can specify
        both hotel and city in a single utterance.
        Alternatively, the user can fill one field at a time.
      </p>

      <h2>Hotel Picker</h2>
      <p>This voice-enabled application lets you pick a
        city and a hotel.
      </p>
      <form id="visual_city_hotel" method="post"
        action="cgi/hotel.pl"
ev:event="focus" ev:handler="#voice_city_hotel" >
        <p id="please_choose">
          Please choose a city and hotel where you wish to
          stay.
        </p>

        <input name="city" type="text"
          ev:event="focus" ev:handler="#voice_city"/>
        <input name="hotel" type="text"
          ev:event="focus" ev:handler="#voice_hotel"/>

        <input type="submit" value="Submit" />
        <input type="reset" value="Reset"
ev:event="click" ev:handler="#voice_cancel"/>
      </form>
    </body>
  </html>
>
```



# Modularize Formatting Objects

## Learn how to create flexible, reusable Formatting Objects stylesheets

**T**he Extensible Stylesheet Language (XSL) W3C recommendation was created as a means to display XML data. The recommendation includes a transformation language (XSLT) and formatting object (or output format) language (XSL-FO), which together provide the XSL stylesheet developer with the tools necessary to present XML.

The XSL-FO language is like HTML on steroids, because it allows not only cascading stylesheet (CSS)-type functionality, but also pagination and page layout, which are not available using HTML alone (at least not to as great a degree as with XSL-FO). However, such robustness comes at a cost; the language is very (very) verbose, which makes writing XSL-FO XSL stylesheets very difficult; thus, XSL-FO stylesheets are supposed to be produced as the result of an XSL transformation. But someone has to write the XSL stylesheet that transforms the XML into XSL-FO, and that's where this article can help.

The purpose of this article is to show you, the XSL stylesheet developer, how to make XSL-FO XSL stylesheets that are flexible and reusable. In other words, I'm going to show you how to make modularized XSL-FO XSL stylesheets.

Many of the key concepts presented in this article are not XSL-FO specific and can be leveraged to write any kind of XSL stylesheet. However, I concentrated on XSL-FO XSL stylesheets because their verbosity highlights the need for modularization more than most stylesheets.

I'm assuming a working knowledge of both XSLT and XSL-FO. For an introduction, you can refer to many resources on the Internet, including my

article in the January 2002 edition of *XML-Journal*.

### Overview of the Solution

Although this modularization solution is very flexible, it is also very simple and uses the XSL `<xsl:import>` and `<xsl:attribute-set>` elements, along with the `xsl:use-attribute-sets` attribute. When used properly, these elements and this attribute not only allow you to externalize the way an XML document is transformed into an XSL-FO document (`<xsl:import>`), but also the look-and-feel of the eventual FO document (`<xsl:attribute-set>`, `xsl:use-attribute-sets`). Figure 1 depicts all the pieces of these concepts.

From a high level, the XSLT engine reads in an XML document and XSL stylesheet (nothing new there). The main stylesheet, the one that is read by the XSLT engine, then uses `<xsl:import>` to use two additional stylesheets – one for the layout of the XSL-FO document and another for the “look-and-feel.” The layout document then uses the `use-attribute-sets` attribute to get the “look-and-feel” (i.e., element attribution information) data from the “look-and-feel” XSL stylesheet, which is implemented as a series of `<xsl:attribute-set>` elements.

With the layout and look-and-feel externalized and separated into different stylesheets, the stylesheet writer can easily exchange both/either the layout and/or look-and-feel of the XSL-FO document information. If the XSL-FO document has such flexibility, it follows that the layout and look-and-feel of the eventual output document, a PDF file for example, has equal flexibility.

### A Presentation Example

The example I'm going to present is a good case for the advantages of using a

modular design in XSL. In presentation graphics (e.g., Microsoft PowerPoint), there are templates that can change the layout of pages and the color/graphics scheme of your presentation. I am going to replicate, to some degree, that same kind of functionality using XSL. If you use the ideas in this example, you can impress your friends by creating PDF presentations that are cross-platform with data separated from the presentation (which implies that the same data could potentially be used in entirely different formats).

#### The XML file

The data for the sample is represented in a simple XML file, named `presentation.xml`, that depicts the idea of a presentation with pages containing concepts (such a format is probably a little too coupled to the presentation, but it'll work). The root element of the document is “Presentation,” with descriptive child elements called “Title,” “Author,” “Date,” and “CorporateLogo,” and a series of “Page” elements that take the following form:

```
<Page style="bullet">
  <Position>01</Position>
  <Title>Title Text</Title>
  <Concept>Some concept</Concept>
  ...more concepts
</Page>
```

The “style” attribute is referred to in the layout stylesheet to create the proper text layout in the presentation (e.g., PDF) output. The example is bullet, but you could extend the idea to be graphic, column, bullet-graphic, or any other style you require. You'll see shortly how the value of this attribute is used.

#### AUTHOR BIO

Frank Neugebauer is a member of the Business Consulting Services division of IBM Global Services specializing in distributed solutions based on IBM's Insurance Applications Architecture (IAA). He has contributed several articles to leading industry publications and served the role of technical editor for the Pearson Technology Group books *Teach Yourself XML in 21 Days* and *XSL Formatting Objects Developer's Handbook*.

The position element allows you to put the “Page” elements in any order within the XML document and still have them laid out the way you want in the final output. Like the “style” attribute, the “Position” element is used by the layout stylesheet.

### The main XSL stylesheet

Ultimately, the XSLT engine requires the name of a stylesheet to be processed. In this example, that file is called `main.xml` and is very simple since its main purpose is to point to the correct layout and look-and-feel stylesheet. Listing 1 shows the contents of that stylesheet, minus the XML declaration and `<xsl:stylesheet>` element.

Within Listing 1, the `<xsl:import>` elements point to the layout and look-and-feel stylesheet, respectively. To change the template, you simply create a new layout and/or look-and-feel stylesheet and change the value of the import.

The other interesting point of the `main.xml` stylesheet is simply that it creates the `<fo:root>` element and uses the `<xsl:apply-templates>` to transfer processing to other templates (which are defined in the `plainPageLayout.xml` stylesheet).

### Look-and-feel

The `lookandfeel.xml` stylesheet contains a number of `<xsl:attribute-set>` definitions, which can be referenced using the `xsl:use-attribute-sets` property with an FO element. The contents of each `<xsl:attribute-set>` are the equivalent of attributes on XSL elements.

Take, for instance, a relatively simple `<fo:simple-page-master>` element, which defines a template for a page.

```
<fo:simple-page-master
  master-name="Title"
  margin-top="1in"
  margin-bottom="1in"
  margin-left="1in"
  margin-right="1in"
  page-height="8.5in"
  page-width="11in"
>

<fo:region-body/>
<fo:region-after
  extent="1in"/>
</fo:simple-page-master>
```

You can externalize all the attributes of the `<fo:simple-page-master>` element into the `lookandfeel.xml` file by defining an `<xsl:attribute-set>` element as follows:

```
<xsl:attribute-set name="titlePage
  Layout">
  <xsl:attribute name="margin-
    top">3in</xsl:attribute>
  <xsl:attribute name="margin-
    bottom">1in</xsl:attribute>
  <xsl:attribute name="margin-
    left">1in</xsl:attribute>
  <xsl:attribute name="margin-
    right">1in</xsl:attribute>
  <xsl:attribute name="page-
    height">8.5in</xsl:attribute>
  <xsl:attribute name="page-
    width">11in</xsl:attribute>
</xsl:attribute-set>
```

Then, you can refer to the attribute set within the `<fo:simple-page-master>` by using the “`xsl:use-attribute-sets`” attribute, such as:

```
<fo:simple-page-master
  master-name="Title"
  xsl:use-attribute-sets="titlePage-
  Layout">
```

Not only is the syntax easier to read, but the attributes values themselves are now externalized from the layout in what’s essentially the FO equivalent of an external CSS.

### Page layout

In XSL-FO, layout is a matter of setting up page definitions and their contents. In my example, I use the page layout XSL document, named `plainPageLayout.xml`, to perform the template matching on the XML document contents. I did this because the layout is responsible for understanding the output document, something that requires knowledge of the XML input, since the data from the latter becomes part of the former.

The `plainPageLayout.xml` document itself looks like most XSL-FO stylesheets. However, instead of very long attribute listings (for look-and-feel type attributes), there are several `xsl:use-attribute-sets` attributes.

In terms of XSLT, `plainPageLayout.xml` contains three templates: one each to process the “Presentation,” “Page,” and “Concept” elements, respectively. That presentation template creates the basic layout of the resulting XSL-FO document and then creates the title page, using data from the XML document. Throughout this stylesheet, `xsl:use-attribute-sets` attributes are used within `<fo>` elements to retrieve attributes from `lookandfeel.xml`.

The Page template actually processes Page elements by style. That is, the

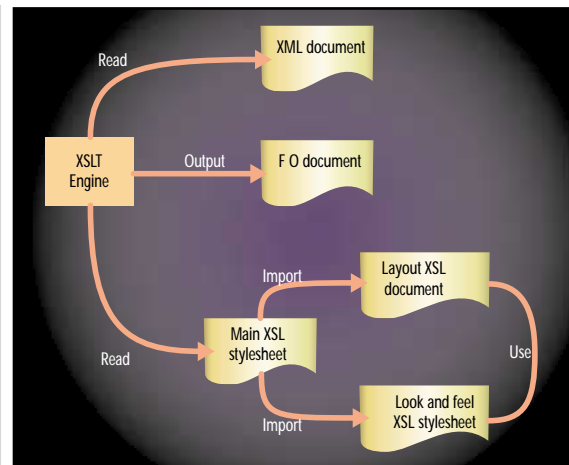


Figure 1 • The modular pieces of the solution

```
<xsl:template
  match="Page[@style='bullet']">
```

element will catch the “bullet” style pages only. Within the template, the non-title pages are set up with a title and two blank lines, after which the “Concept” elements are selected within a `<xsl:apply templates select="Concept"/>` statement. Finally, the “Concept” template processes each “Concept,” creating an XSL-FO list with bullets.

As a side note, the stylesheet can be made even more modularized if you were to externalize your “Page” and/or “Concept” templates. Doing so would let you use the “bullet” style across different layout XSL documents. I did not do so for the sake of simplicity.

...you can impress your friends by creating PDF presentations that are cross-platform with data separated from the presentation

Listing 2 shows the entire contents of `plainPageLayout.xml`. Within Listing 2, notice the comment that reads “`<!-- Shows how to override attributes. -->`.” What I’m showing at that point in the stylesheet is that even if an XSL-FO element uses an attribute set, you can still override and/or amend the attribute set within the element itself, much in the same way you can override/add to an

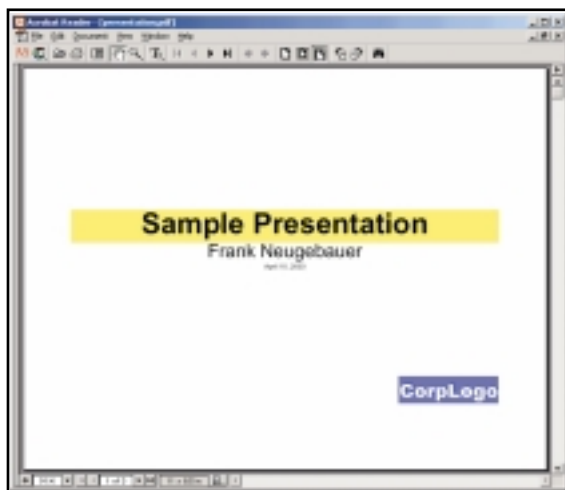


Figure 2 • The main title page of the PDF result

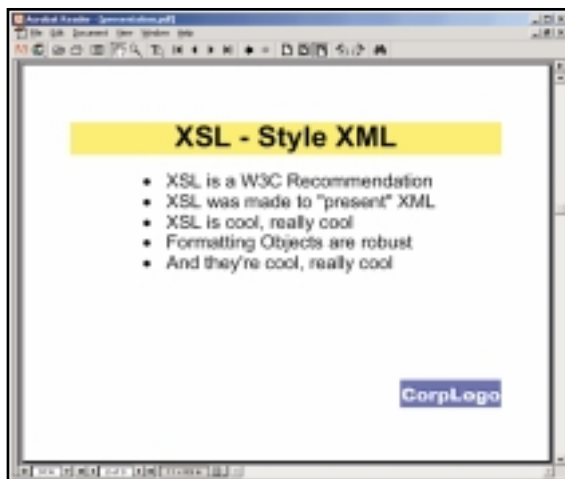


Figure 3 • The first non-title page of the PDF output

HTML element if a CSS is defined – further evidence of the CSS-like nature of attribute sets.

### The result

I believe that the proof of any technology is in the pudding. When you run presentation.xml and main.xml through an FO processor (I used FOP 0.20.3), you will end up with a three-page PDF (if you choose the PDF output option within FOP). Figures 2 and 3 show the first two pages. Page 3 is just like page 2 but contains different concepts (refer to presentation.xml).

Figure 2 shows not only the main “Presentation” element (and the relevant non-page children of “Presentation”), but it also shows some of the XSL-FO design I created. For instance, the logo and yellow background color. The color and font characteristics are externalized within lookandfeel.xml, of course.

Figure 3 uses the same basic layout characteristics of the title page, but also

contains the list of concepts. Every “Page” that uses the “bullet” style will be rendered just as Figure 3 (refer to the third page of presentation.pdf if you download the source code, available at [www.sys-con.com/xml/sourcec.cfm](http://www.sys-con.com/xml/sourcec.cfm)).

### Conclusion

XSL-FO XSL stylesheets are long and difficult to read and write. Furthermore, it's not easy to change layout and look-and-feel without additional modularization. Using the XSL `<xsl:import>` and `<xsl:attribute-sets>` elements along with

the `xsl:use-attribute-sets` attribute can go a very long way toward making such modularization a reality. ☺

### Resources

- XSL: [www.w3.org/Style/XSL](http://www.w3.org/Style/XSL)
- Neugebauer, Frank. “XSL Formatting Objects: Here Today, Huge Tomorrow.” (2002). *XML-Journal*. January: [www.sys-con.com/xml/article.cfm?id=324](http://www.sys-con.com/xml/article.cfm?id=324)

NEUGGS@NETSCAPE.NET

#### LISTING 1 • The main.xml stylesheet

```
<!--
-- XSL Imports. -->
<xsl:import href="plainPageLayout.xml"/>
<xsl:import href="lookandfeel.xml"/>

<xsl:output
  method="XML"
  indent="yes" />

<xsl:strip-space elements="*" />

<xsl:template match="/">
  <fo:root
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:apply-templates/>
  </fo:root>
</xsl:template>
```

#### LISTING 2 • The plainPageLayout.xml document

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
  xmlns:xalan="http://xml.apache.org/xslt"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <xsl:strip-space elements="*" />

  <xsl:template match="Presentation">
    <fo:layout-master-set>
      <fo:simple-page-master
        master-name="Title"
        xsl:use-attribute-sets="titlePageLayout">
        <fo:region-body/>
        <fo:region-after
          extent="lin"/>
      </fo:simple-page-master>

      <fo:simple-page-master
        master-name="Page"
        xsl:use-attribute-sets="otherPageLayout">
        <fo:region-before extent="lin"/>
        <fo:region-body margin-top=".75in"/>
        <fo:region-after extent=".75in"/>
      </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence
      master-reference="Title">
      <fo:static-content
        flow-name="xsl-region-after">

        <fo:block text-align="right">
          <fo:external-graphic
            src="{CorporateLogo}"
            height="auto"
            width="auto"/>
        </fo:block>
```



```

</fo:static-content>

<fo:flow
  flow-name="xsl-region-body">
  <fo:block
    xsl:use-attribute-sets="mainTitleBlock">
    <xsl:value-of select="Title"/>
    </fo:block>

    <fo:block
      xsl:use-attribute-sets="subTitleBlock">
      <xsl:value-of select="Author"/>
      </fo:block>

    <!-- Shows how to override attributes. -->
    <fo:block
      xsl:use-attribute-sets="plainTextBlock"
      text-align="center">
      <xsl:value-of select="Date"/>
      </fo:block>
    </fo:flow>
  </fo:page-sequence>

  <xsl:apply-templates select="Page">
  <xsl:sort select="Position"/>
  </xsl:apply-templates>
</xsl:template>

<xsl:template match="Page[@style='bullet']">
  <fo:page-sequence
    master-reference="Page">
    <fo:static-content
      flow-name="xsl-region-after">
      <fo:block text-align="right">
        <fo:external-graphic
          src="{../CorporateLogo}"
          height="auto"

```

```

width="auto"/>
</fo:block>
</fo:static-content>

<fo:flow
  flow-name="xsl-region-body">
  <fo:block
    xsl:use-attribute-sets="titleBlock">
    <xsl:value-of select="Title"/>
    </fo:block>

    <fo:block color="white">space</fo:block>
    <fo:block color="white">space</fo:block>

    <xsl:apply-templates select="Concept"/>
  </fo:flow>
</fo:page-sequence>
</xsl:template>

<xsl:template match="Concept">
  <fo:list-block
    xsl:use-attribute-sets="bullets">
    <fo:list-item>
      <fo:list-item-label end-indent="label-end()">
        <fo:block>
          <fo:inline font-
            family="Symbol">&#x2022;</fo:inline>
          </fo:block>
        </fo:list-item-label>

        <fo:list-item-body start-indent="body-start()">
          <fo:block><xsl:value-of select="."/></fo:block>
        </fo:list-item-body>
      </fo:list-item>
    </fo:list-block>
  </xsl:template>
</xsl:stylesheet>

```

Download the Code  
[www.sys-con.com/xml](http://www.sys-con.com/xml)

## XML-J ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	<a href="http://xmlj.altova.com/wsd/">http://xmlj.altova.com/wsd/</a>	978-816-1600	52
Ektron	<a href="http://www.ektron.com/xml">www.ektron.com/xml</a>	603-594-0249	21
JDJ Workshop	<a href="http://www.sys-con.com/education">www.sys-con.com/education</a>	201-802-3058	31
Linux Business & Technology	<a href="http://www.sys-con.com">www.sys-con.com</a>	888-303-5282	11
Macromedia	<a href="http://www.macromedia.com/go/cfmxad">www.macromedia.com/go/cfmxad</a>	415-252-2000	2
Mindreef	<a href="http://www.mindreef.com">www.mindreef.com</a>	603-465-2204	4
PolarLake	<a href="http://www.polarlake.com">www.polarlake.com</a>		9
PriceWaterHouseCoopers	<a href="http://www.pwcglobal.com/tech-forcast_syscon">www.pwcglobal.com/tech-forcast_syscon</a>		6
Quest Software	<a href="http://java.quest.com/jjsc/ws">http://java.quest.com/jjsc/ws</a>	800-663-4723	51
SYS-CON Media	<a href="http://www.sys-con.com/2001/sub.cfm">www.sys-con.com/2001/sub.cfm</a>	888-303-5282	49
SYS-CON Reprints	<a href="http://www.sys-con.com">www.sys-con.com</a>	201-802-3026	45

**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. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

## Once you're in it...

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

## reprint it...



Contact Carrie Gebert  
 201 802-3026  
[carrieg@sys-con.com](mailto:carrieg@sys-con.com)

SYS-CON  
 MEDIA

# Reprints

# MathML

WRITTEN BY AYESHA MALIK



## Enabling mathematical functionality for the Web

**M**athML is a World Wide Web Consortium (W3C) standard that has been created so that mathematical expressions can be displayed, manipulated, and shared over the Web. According to the W3C, "The goal of MathML is to enable mathematics to be served, received, and processed on the World Wide Web, just as HTML has enabled this functionality for text."

The MathML API can be found at the W3C Web site ([www.w3.org](http://www.w3.org)). It has been designed as an XML application and provides two main sets of tags: one for the visual presentation of mathematics and the other for the content in equations. The W3C Math Working Group states that MathML should be created and edited using specialized tools and is not designed to be written primarily by hand, given the length and complexity of its API.

### Benefits of MathML

The introduction of MathML brings several benefits, including:

- **Computation:** Given that MathML includes tags for the content of the mathematics, it can be used for computation (note that this is different from GIF or PDF files where mathematical constructs can be represented but can't be manipulated). In fact, major computer algebra systems, such as Mathematica, have begun to support cut-and-paste of MathML into their documents, enabling their users to transfer math from Web pages and other sources into their documents. After the MathML document has been transferred to a mathematical tool, it can be manipulated for calculation, graphing, searching, and analysis.
- **Public standard:** The fact that MathML is a public standard paves the way for

the creation of tools and solutions from a number of vendors and open-source organizations. It also means that mathematical data can be easily shared among different organizations and in this way facilitates research and information sharing (both of which are the goals of the World Wide Web).

- **XML advantages:** MathML is based on XML so it brings all the advantages of XML with it. It can be used for representation on any application that handles XML, including Web, paper, and PDF. Also, the representation can be changed using a stylesheet document.

### Overview

MathML's API consists of two main groups: presentation elements and content elements. Presentation elements describe a mathematical notation's visually oriented two-dimensional structure while content elements describe what the math means.

### Presentation Markup

MathML has about 30 presentation elements that accept 50 attributes. Most elements represent templates or patterns for laying out subexpressions.

#### Elements

Presentation elements can be broadly grouped under the following five classes of elements.

1. **Token elements:** Token elements represent the most basic structures in mathematics. For example:
  - **<mi>:** Identifier, such as a variable, function name, constant
  - **<mo>:** Operator, such as a summation
  - **<mn>:** Number

There are also a few presentation ele-

ments that are empty elements, and are used mostly in connection with alignment.

2. **General layout:** General layout elements describe the nature of the layout. For example:

- **<mfrac>:** Form a fraction from two subexpressions
- **<msqrt>:** Form a square root sign

3. **Scripts and limits:** Scripts and embellishments to symbols are common in mathematical notation. The elements in this section address this requirement. For example:

- **<msub>:** Attach a subscript to a base
- **<msup>:** Attach a superscript to a base

4. **Tables:** Matrices, arrays, and table-like mathematical notations are also represented using tags in MathML. For example:

- **<mtable>:** Table or matrix
- **<mtr>:** Row in a table or matrix

5. **Actions:** The maction element is in a category by itself, and allows coding of various kinds of actions on notation, such as occur in an expression which toggles between two pieces of notation.

- **<maction>:** Binds actions to a subexpression

It is important to note that the order of elements is important in MathML. This is the only way that the MathML processor can understand the mathematical constructs. For example, the first child of an mfrac element is the numerator and the second child is the denominator.

#### Attributes

Presentation elements can take up to 50 different attributes. Attributes generally

### AUTHOR BIO

Ayesha Malik is a senior software consultant at Object Machines and has worked extensively on large Java, XML, and Web services systems in a wide range of industrial environments. She is the author of articles on software development, and has spoken at several industry conferences. Ayesha holds a BA with honors from Harvard University and an MS from Columbia University, where she studied operations research, applied mathematics, and computer science.

specify additional optional information about the element. Each attribute has a name and a value. Using attributes, it's possible to precisely control how an expression will look when displayed. For example, the `mfrac` element has an attribute called `linethickness` and is described below:

```
attribute: linethickness
values: number [ v-unit ] | thin |
medium | thick
default: 1 (rule thickness)
```

## Content Markup

There are about 120 content elements and they accept close to a dozen attributes. Content markup facilitates applications other than display, like computer algebra and speech synthesis. The scope of content markup includes arithmetic, algebra, logic, relations, set theory, calculus, sequences, series, functions, statistics, linear algebra, and vector calculus.

### Elements

There are many elements in the content markup part of MathML. The best way to think about elements is to categorize them into six buckets as follows.

1. **Containers:** Container elements are used to indicate the basic units of mathematical content such as mathematical identifiers, numbers, and symbols. For example:

- **<cn>:** Used to represent numbers
- **<ci>:** Used to construct an identifier or variable

2. **Operators, qualifiers, and functions:** These are tags used to define different mathematical and statistical functions and operators. For example:

- **<sin>:** Function for sine
- **<divide>:** Operator for division
- **<lowlimit>:** Qualifier

3. **Relations:** Relations are characterized by the fact that, if an external application were to evaluate them, they would typically return a truth value. For example:

- **<eq>:** Equal
- **<neq>:** Not equal

4. **Conditions:** The `<condition>` element is used to define the "such that" construct in mathematical expressions. The condition element is always used together with one or more `bvar` elements and the interpretation depends on the context.

5. **Syntax and semantics:** This class provides additional information required for mathematical processing. For example:

- **<mappings>:** Semantics, annotation, annotation-xml

6. **Constants and symbols:** These are a collection of predefined constants and symbols which represent frequently encountered concepts. For example:

- **<integers>:** A set of integers

The `<apply>` element is perhaps the single most important content element. It is used to apply a function or operation to a collection of arguments. For example:

```
<mrow>
<apply>
  <minus/>
  <ci>a</ci>
  <ci>b</ci>
</apply>
</mrow>
```

### Attributes

There are about 12 attributes for the content presentation markup in MathML. These are used to add information about the content. For example:

```
<cn type="real"> 12345.7 </cn>
```

## Creating MathML

There are a few Web browsers that support MathML, including Netscape 7.0, Amaya, and Mozilla. Amaya is the W3C's editor and browser and is used to demonstrate and test many of the new developments in Web protocols and data formats. In the simple example shown in Listing 1, I will use Amaya. Amaya can be downloaded from [www.w3.org/Amaya/User/BinDist.html](http://www.w3.org/Amaya/User/BinDist.html).

Here we have the normal structure of an XHTML document. It begins with the start tag `<html>` embellished with an XML namespace declaration and language assertions. A head element contains a title as is customary. Then the `<body>` beginning also has a namespace declaration of an abbreviative prefix letter `m` to be used for the standard MathML namespace. Next comes a simple paragraph. Finally we get the math element which also has a namespace association declared. Inside the math element is MathML markup as we are beginning to be used to it. It is rendered in Amaya as shown in Figure 1

Amaya and Mathematica are two examples of editors that can be used for editing the content of MathML. The W3C MathML page also mentions a LaTeX to MathML converter known as WeM, which is a MathML editor that converts a subset LaTeX to MathML. It can be tested online and is also available for download (GPL, requires PHP). Several other available tools can also



Figure 1 • MathML markup

be found on the MathML Web site ([www.w3.org/Math](http://www.w3.org/Math)).

## Summary

MathML is a powerful XML-based markup language for publishing mathematics on the Web. MathML makes it possible to develop Web-based applications for displaying, searching, indexing, archiving, and evaluating mathematical content. It consists of two basic structures: presentation markup for describing math notation, and content markup for describing mathematical objects and functions. The motivation for MathML is a world in which individuals and companies want to move away from proprietary data formats and use XML to express, interchange, and share information. With increasing tool and browser support (both Microsoft and Netscape have publicly declared support for the XML recommendation), MathML's importance and prevalence will continue to grow.

AYESHA.MALIK@OBJECTMACHINES.COM

### LISTING 1 •

```
<html xmlns="http://www.w3.org/1999/xhtml"
lang="en" xml:lang="en">
<head>
<title>MathML's Hello Square</title>
</head>

<body>

<p> This is a perfect square:</p>

<math
xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <msup>
      <mfenced>
        <mrow>
          <mi>a</mi>
          <mo>+</mo>
          <mi>b</mi>
        </mrow>
      </mfenced>
      <mn>2</mn>
    </msup>
  </mrow>
</math>

</body>
</html>
```

▼ Download the Code  
www.sys-con.com/xml

## Novell Helps Developers Bridge the Gap Between Web Services and Directories

(Scottsdale, AZ) – Novell has announced that Novell eDirectory now supports the Directory Services Markup Language (DSML v2) standard, which provides access to eDirectory using common Web services protocols. Novell has also released its new DSML code to the open-source community.

DSML bridges the gap between developing to Web



services and developing to a directory. With it, traditional LDAP directory operations and their results can be expressed in XML; so application developers who are familiar with developing using XML and SOAP can now add directory functionality to their applications using tools they already know.

[www.novell.com](http://www.novell.com)

## Sun Labs Contributes XACML Implementation to Open-Source Project

(Santa Clara, CA) – Sun Microsystems, Inc., has announced the release of its implementation of the new XACML OASIS Open Standard for security under an open-source license. XACML contributes to the simplification and cost reduction of developing and deploying secure Web services – or any application that requires secure access control.

Sun's XACML Implementa-

tion was developed by the Internet Security Research Group (ISRG) within Sun Microsystems Laboratories, and could have far-reaching impact on enterprise security as well as developer productivity. As XACML replaces the current patchwork of proprietary access-control policy languages, administrators will no longer need to learn these many languages and translate policies between them, and software developers won't have to invent their own languages and write custom code to support them as they do today. [www.sun.com](http://www.sun.com)

## Microsoft Announces Windows RMS for Windows Server 2003

(Redmond, WA) – Microsoft has announced plans for Windows Rights Management Services (RMS), a new technology for Windows Server 2003 that will give organizations advanced ways to help secure sensitive internal business information including financial reports and confidential planning documents. Windows RMS will work with applications to provide a platform-based approach to providing persistent policy rights for Web content and sensitive corporate documents of all types. Beta code will be broadly available in the second quarter of 2003.

RMS technology uses XrML (Extensible Rights Markup Language), an emerging standard for the expression of rights on digital content. Microsoft will release two software development kits in the second quarter

of 2003 to enable developers to begin to build rights management capabilities into a broad range of intra-enterprise solutions and applications for Windows clients.

[www.microsoft.com](http://www.microsoft.com)

## RosettaNet Announces Completion of Software Interoperability Trials

(Santa Ana, CA) – RosettaNet has announced the successful completion of the RosettaNet Software Interoperability Trials.

The Software Interoperability Trials tested RosettaNet Implementation Framework (RNIF) connectivity software from 10 different vendors, including Cyclone Commerce, Fujitsu Limited, GridNode,



Global eXchange Services, iSoft, Oracle Corp., Sterling Commerce, TIBCO Software, Vitria, and webMethods, Inc. Drake Certivo, Inc., a vendor-neutral third party specializing in compliance and interoperability testing, administered the RosettaNet trials.

[www.rosettanet.org](http://www.rosettanet.org)

## OASIS Members Advance Reliable Message Delivery for Web Services

(Boston) – Members of the OASIS standards consortium announced plans to collaborate on the development of a generic and open model for ensuring reliable message delivery for Web services. The new OASIS Web Services Reliable Messaging (WS-RM) Technical Com-

mittee will work to establish a standard, interoperable way of achieving reliability at the SOAP messaging level and potentially with other messaging protocols.

The OASIS Reliable Messaging specification will address



message persistence, acknowledgement, and resending; elimination of duplicate messages; ordered delivery; and delivery status awareness for sender and receiver applications. It will provide WSDL definitions for reliable messaging, and the message formats will be specified as SOAP headers and/or body content.

[www.oasis-open.org](http://www.oasis-open.org)

## PowerBuilder 9.0 Powers RAD Across Leading Technologies

(Dublin, CA) – A wealth of enhancements and productivity-boosting features distinguish the newly announced Sybase PowerBuilder 9.0. This version of Sybase's RAD tool enables developers to continue building rich-client applications for their mission-critical business requirements, and at the same time pro-



pel them further into Web and n-tier development with significant new capabilities.

New in PowerBuilder 9.0 is the XML DataWindow, which imports data directly from an XML document and saves data that was retrieved from any data source as a fully customized XML document, or as a document or string using XSL Formatting Objects (XSL-FO) or PDF. In addition, the PowerBuilder Document Object Model defines how XML documents can be accessed and manipulated.

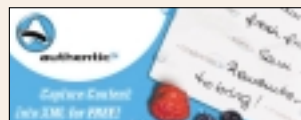
[www.sybase.com](http://www.sybase.com)

## Altova's AUTHENTIC 5 to Be Included with Software AG's Tamino Server – Free of Charge

(Beverly, MA and Darmstadt, Germany) – Altova, Inc., and Software AG have announced that AUTHENTIC 5 will be included as a standard component of all future Tamino Server product offerings at no additional cost to the Tamino Server customer.

Software AG's Tamino XML

Server is a high-performance platform for storing, publishing, and exchanging XML documents. Altova's AUTHENTIC 5 is a browser-enabled XML document editor. The AUTHENTIC 5



word-processor style interface is ideally suited as a lightweight editing component for building document frameworks, standards-based XML document management systems for Web publishing, or knowledge management applications.

[www.altova.com](http://www.altova.com), [www.soft  
wareag.com](http://www.softwareag.com)



# SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

## AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!



**TO  
ORDER**

•Choose the Multi-Pack you want to order by checking next to it below. •Check the number of years you want to order. •Indicate your location by checking either U.S., Canada/Mexico or International. •Then choose which magazines you want to include with your Multi-Pack order.

**Pick a 3-Pack, a 6-Pack or a 9-Pack**

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.

### ☐ Linux Business & Technology

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 /	Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

### ☐ Java Developer's Journal

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

### ☐ Web Services Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

### ☐ .NET Developer's Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

### ☐ XML-Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

### ☐ WebLogic Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

### ☐ ColdFusion Developer's Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

### ☐ Wireless Business & Technology

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$192	You Pay: \$139 /	Save: \$53 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$79.99 /	Save: \$16
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

### ☐ WebSphere Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

### ☐ PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

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

## 3-Pack

Pick any 3 of our  
magazines and save  
up to **\$275<sup>00</sup>**  
Pay only \$175 for a  
1 year subscription  
plus a **FREE CD**

- 2 Year - \$299.00
- Canada/Mexico - \$245.00
- International - \$315.00

## 6-Pack

Pick any 6 of our  
magazines and save  
up to **\$350<sup>00</sup>**  
Pay only \$395 for a  
1 year subscription  
plus 2 **FREE CDs**

- 2 Year - \$669.00
- Canada/Mexico - \$555.00
- International - \$710.00

## 9-Pack

Pick 9 of our  
magazines and save  
up to **\$400<sup>00</sup>**  
Pay only \$495 for a  
1 year subscription  
plus 3 **FREE CDs**

- 2 Year - \$839.00
- Canada/Mexico - \$695.00
- International - \$890.00

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**Subscribe Online Today [www.sys-con.com/2001/sub.cfm](http://www.sys-con.com/2001/sub.cfm)**

**SYS-CON  
MEDIA**

# Moving Toward Convergence

A few years ago at an early XML conference, an attendee made the point that XML was such a useful technology for data portability that it would eventually become ubiquitous – part of every tool, server, and application. He went on to predict that XML would become so commonplace that the idea of attending an XML conference would eventually seem silly. “In a few years, a conference on XML will seem as ridiculous as a conference on ASCII would today,” he quipped.



WRITTEN BY  
JOHN MAGEE

He was on the right track, but by focusing on XML only as a way to move data between proprietary applications even this bullish prognosticator was quite likely understating XML's long-term potential to improve the capabilities of information systems.

XML has been a smashing success. It's widely used in publishing as a lightweight alternative to SGML for separating form and content. It plays a key role in messaging and business-to-business data exchange, offering a low-cost replacement for technologies such as EDI. Its on-the-fly transformation capabilities make it ideal for personalized user interfaces on non traditional computing devices. It makes software more interoperable by providing a language-independent foundation for defining vendor-neutral interfaces, such as OASIS's WSRP portlet initiative and JSR 172 for wireless devices. XML is also the key technology that makes Web services stand out from previous generations of distributed computing models. And XML is well on its way toward ubiquity. No self-respecting developer today even thinks of defining a new interface or config file without turning to XML, and some level of XML support has crept into just about every product in every category of software.

As significant as these contributions are, however, their impact pales in comparison to the longer-term change that XML will help bring about, which involves a fundamental shift in how we manage, access, and share information. The schism between the document-centric and data-centric camps actually highlights the source of what will be XML's most lasting contribution: its ability to bridge the worlds of structured and unstructured data.

Today less than 10 percent of the world's information is managed as the valuable resource that it is, and most of that is traditional “rows and columns” structured data. Being able to manage – to capture, store, index, search, analyze, and share – more of the remaining data will be a Very Big Deal. XML will make this possible by enabling the conversion of unstructured data into semi-structured data that can then be actively managed, both on its own and in hybrid applications that require structured and unstructured information. The canonical but prosaic example of catalog information, which combines structured data such as part numbers and prices with unstructured content such as images and text, only hints at the potential for knowledge and efficiency when we have information systems that can truly manage all our data.

When unstructured information becomes a managed resource, it can be integrated into more organizational processes. It can be analyzed with data mining and business intelligence tools. Users can search across information that was previously stored in silos, such as file systems, document repositories, Web sites, and e-mail. Automating collaborative processes becomes easier. And capturing information as it is created becomes the norm. Consider the enormous amount of business information that resides in spreadsheets today. As these documents and the tools that create them become XML-enabled, it will be easier to capture that data in repositories where it can be made immediately available to applications for processing.

XML's flexibility and applicability to a wide range of data management problems allow the industry to progress toward convergence on several fronts, without the need for a “big bang” paradigm shift. Document-intensive industries are benefiting from standardizing their document formats with XML, and Schema continues to gain momentum as the preferred way of defining the structure of complex documents. Tool vendors are XML-enabling content-creation tools to make it easier to capture information in content repositories. Server vendors are XML-enabling other infrastructure software to make it easier to share and repurpose XML-based information. These incremental steps toward convergence represent an organic process that is more likely to happen in the real world, where investments in new technology must deliver immediate tangible business benefits.

XML standards and XML-enabled tools will not, by themselves, turn data into information and eliminate the problems caused by silos of information stored in different formats. To realize the full benefits of convergence, organizations need to rethink how they store and manage information. An architecture based on fewer integrated information repositories that provides cost-effective scalability, security, and manageability will be key.

Such a repository will need to provide support for key XML standards such as Schema and DOM, and support for navigation-based access via XPath, as well as highly optimized querying through SQL, XQuery, and SQL XML (SQL/X). To support as many tools and other clients as possible, it should provide direct access from a variety of protocols including HTTP, FTP, and WebDAV.

Because progress toward convergence is happening in an incremental way, it's possible to misinterpret these advances as less significant than they really are. But at a certain point a difference of degree becomes a difference of kind, and we are left with a very different world than the one we started with. At Oracle, we are continuing to leverage XML's capabilities in a wide range of ongoing product releases and R&D efforts, on the firm conviction that today's information systems represent only a glimpse of what will be possible tomorrow. ☐

JOHN.MAGEE@ORACLE.COM

## AUTHOR BIO

John Magee is vice president of Oracle9i at Oracle. John has over 14 years of experience in the enterprise software industry and has held positions in product development, product management, and product marketing. In his current role, he manages technical product marketing for Oracle's application server and development tools products, and is responsible for evangelizing Oracle technology initiatives around J2EE, XML, and Web services.

# Quest Software

<http://java.quest.com/jcs/ws>

# Altova

<http://xmlj.altova.com/wsdl>