

XML JOURNAL

THE ULTIMATE XML ENTERPRISE RESOURCE

August 2003 Volume: 4 Issue:8

xml-journal.com



Conference Program page 31

From the Editor

Revisiting RSS

Enjoying 'Really Simple Syndication'
by Hitesh Seth pg. 3

Guest Editorial

XML in the Real Real World
Look beyond the buzzwords
by Tim Bray pg. 5

CMS

Content Management,
XML, and the Promise of
Web Services

How optimizing XML
can make your life simpler
by Bill Rogers pg. 26

Show Report

JavaOne 2003

Focus on XML and Web services
by Chris Peltz pg. 42

DISPLAY UNTIL OCTOBER 31, 2003

\$6.99US \$7.99CAN



0 09281 01314 3

SYS-CON
MEDIA

Introducing Microsoft InfoPath 2003

Part 2 Extending InfoPath solutions with BizTalk Server 2004

Feature: Moving from Web Services to an Enterprise Service-Based Architecture

Act now to expand your business capabilities



Bob Zurek

6

Feature: Introducing Microsoft InfoPath 2003

Part 2 Extending InfoPath solutions with BizTalk Server 2004

Thom Robbins

10

XMLSPY: Aggregation with XMLSPY

The advantages of XML... without all the <angle brackets>

Andrew Solymosi

14

XForms: Hands-on XForms

An essential
guide to simplifying the creation and management of XML information

Micah Dubinko

18

Feature: Object-Oriented XSLT

OOX brings a new paradigm for content management



Pietro Michelucci

20

XSLT: Using XSLT to Generate SQL

A simple, adaptable example to get you started

Greg Watson

28

Feature: Multipass Validation with

XSD and Schematron Part 2 Using Schematron

Eric J. Schwarzenbach

& David Kershaw

rules to enforce good W3C schema design – an exercise for developers

38

IBM

ibm.com/developerWorks/toolbox/seeit

FOUNDING EDITOR

Ajit Sagar ajit@sys-con.com

EDITORIAL ADVISORY BOARD

Graham Glass graham@themindelectric.com

Coco Jaenicke cjaenicke@attbi.com

Sean McGrath sean.mcgrath@propylon.com

Simeon Simeonov talktosim@polarisventures.com

EDITORIAL

Editor-in-Chief

Hitesh Seth hitesh@sys-con.com

Editorial Director

Jeremy Geelan jeremy@sys-con.com

Managing Editor

Jennifer Van Winckel jennifer@sys-con.com

Editor

Nancy Valentine nancy@sys-con.com

Associate Editors

John Evdemon jevdemon@sys-con.com

Jamie Matusow jamie@sys-con.com

Gail Schultz gail@sys-con.com

Jean Cassidy jean@sys-con.com

PRODUCTION

Production Consultant

Jim Morgan jim@sys-con.com

Art Director

Alex Botero alex@sys-con.com

Associate Art Directors

Louis F. Cuffari louis@sys-con.com

Richard Silverberg richards@sys-con.com

Assistant Art Director

Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Tim Bray, Micah Dubinko, David Kershaw,

Pietro Michelucci, Chris Peltz,

Thom Robbins, Bill Rogers,

Eric J. Schwarzenbach, Hitesh Seth,

Andrew Solymosi, Greg Watson, Bob Zurek

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.

Worldwide Newsstand Distribution

Curtis Circulation Company, New Milford, NJ

©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.



Revisiting RSS

WRITTEN BY HITESH SETH



I'm one of those technology enthusiasts who like to be on the edge, which means that if I'm not creating news, I at least like to read a lot to keep up with the rapidly changing world of technology. A part of my morning (and sometimes even my night) is spent taking a good look at some of the popular technology Web sites. I think I look through at least 20 such sites each day, in most cases browsing through the headlines and in some cases diving deep into the articles.

In a recent search I came across an article by Dare Obasanjo on MSDN, titled "RSS Bandit." In his article, Dare talks about having utilized the various XML capabilities available in the .NET Framework to develop a desktop-based news reader that retrieves news from different Web sites and puts it together in a nice desktop application, triggering the user for new content. I downloaded the application and felt that it would meet most of my needs. I did use



it for a while, but then I came to the conclusion that as a die-hard Web user, I would rather have similar functionality in a Web application. This would help me keep the application on one central "server" (or even my own desktop, given the fact that I am actually running Windows 2003 Server) and run it within a familiar interface, a desktop browser. Running it on the server would also enable it to be used from devices other than my PC. I would just need to ensure that the portal framework I'm using and my RSS portlet are mobile.

If you haven't come across RSS yet, RSS is "Really Simple Syndication." Simply put, RSS provides a simple way for Web sites to provide syndicated content to downstream applications. Of course, RSS is based on XML and it's not really a new development. In fact, it dates all the way back to 1999, when RSS was introduced by Netscape to add channels to their my.netscape.com portal. Since then RSS has become almost the de facto standard, particu-

larly for news sites to provide headlines. After using RSS Bandit and my own portal application, I no longer enjoy news sites that don't provide RSS-based headlines.

RSS can be processed in multiple ways. I prefer to use XSLT to transform RSS content into HTML within a portal environment. I'm a huge proponent of Web-based and portal frameworks. So in one single aggregated page, I can get related content – news headlines, contact information, and even weather and stock quotes.

A good thing about open standards is that they can be used in scenarios for which they weren't specified. A nice RSS example is illustrated by Greg Reinacker, who has created a simple ASP.NET application to publish system Event logs as RSS. By the way, Greg has also authored another neat application based on RSS, called NewsGator, that provides RSS-based content within Microsoft Outlook. Also, most weblogs are RSS-enabled as well. I'm sure you can think of other content to RSS-enable. So, what are you waiting for? Use an RSS-based aggregator and be prepared for the extra loads of knowledge that you will get by aggregating top news sites, technology sites, and even people's weblogs.

...

In last month's editorial, I gave a brief summary of Microsoft's Tech•Ed conference, from an XML perspective. Right after Microsoft's Tech•Ed was Sun's JavaOne, held in San Francisco, California, June 10–13. Unfortunately, I wasn't able to make it to Java's big event, but luckily Chris Peltz attended and provides a report on JavaOne in this issue of XML-J. If you weren't able to make it to the conference, there's more good news. Most of the sessions are now available online at Sun's JavaOne site (<http://java.sun.com/javaone>). One standard, two competing platforms. That, in a nutshell, is the relationship between XML, Microsoft .NET, and J2EE.

Speaking of conferences, our own Web Services Edge West will be held September 30 – October 2, 2003, in Santa Clara, California. Preparations are in full swing – check out www.sys-con.com/webservicesedge2003west for further details. ☛

AUTHOR BIO

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

HITESH@SYS-CON.COM

Mindreef

www.mindreef.com

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

Director of Sales & Marketing

Megan Mussa megan@sys-con.com

Advertising Sales Manager

Alisa Catalano alisa@sys-con.com

Associate Sales Managers

Carrie Gebert carrieg@sys-con.com

Kristin Kuhnle kristin@sys-con.com

SYS-CON EVENTS

President

Grisha Davida grisha@sys-con.com

Conference Manager

Michael Lynch mike@sys-con.com

Sales Executive, Exhibits

James Donovan james@sys-con.com

CUSTOMER RELATIONS

Circulation Service Coordinators

Niki Panagopoulos niki@sys-con.com

Shelia Dickerson shelia@sys-con.com

Edna Earle Russell edna@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

XML in the Real Real World

WRITTEN BY TIM BRAY



When we at Antarctica start talking to a potential customer or partner, they say, "That's Tim Bray's company? So this is XML-based data visualization?"

And we have to say, "No, it's ordinary database visualization. But because it's modern software, there's a lot of XML in the plumbing." And in this there's a lesson.

A lot of people who care about XML spend a lot of time thinking and worrying about where in the real world of applied technology XML is getting traction.

If you read the trade press and listen to the prognosticators, XML is a landscape of groundbreaking initiatives, industry buzzwords, and architectural upheaval. Here are a few examples:

- The many layers of Web services technology are going to lead to the rebirth of EAI.
- The Semantic Web is going to teach my refrigerator to talk to my local beer vendor when my supply runs low.
- Microsoft's .NET platform is leading to a new dawn of productivity and interoperability for developers.
- Executives and prognosticators alike are telling us that XML will achieve "zero latency" or "real-time business" or "customer focus" or whatever the buzzword of the day is.

But something's wrong with this picture. None of the above has much to do with the way we're using XML, and none seems to have much to do with the way people I talk to in the real worlds of manufacturing or financial services or publishing are using it either.

When I talk to people about how they're really using XML, I detect a few patterns, but they're not the ones I read about in business publications. For example:

- At Antarctica, we wanted to experiment with a Flash client. We had our server, which normally sends out an HTML picture of an information space, send out an XML description, which was pretty easy. Then we programmed the new Flash 6 software to parse the XML and draw the map right on the desktop. That was harder, but it all worked fine; and the Flash maps looked great.
- At a major Web portal, they're assembling their search result and news screens by shooting XML messages off to a half dozen business partners, getting more XML messages back, extracting information from them, and gluing it all together for the portal screen. They didn't write any schemas or

organize any task forces, they just e-mailed examples around until everyone was happy with them, and went to work.

- At Antarctica, we're always receiving new information that a customer (or prospect) wants mapped. We used to get tab-delimited files and Oracle dumps and all sorts of other old-fashioned, awkward, fragile interchange formats. With every month that goes by, more and more people say "...or I could just dump that for you in XML, if you'd like." Of course we'd like. It simplifies the internationalization issues, and when something goes wrong (and something always goes wrong) the XML makes it obvious where the breakage is, so you can fix it.
- The Government of Ireland is working really hard to build a coherent, efficient Web presence for all its citizens. After looking at a dozen different portal architectures, they've decided the way to go is to build some service-oriented infrastructure, define some XML message interchange vocabularies, and get out of the way of the individual departments. It's a bold stroke, and it just might work.

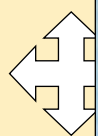
What common threads do you see? I see a tendency to improvise, the application of a lot of ingenuity, and an assumption that anything in the world of computing can be made to talk to anything else in the world of computing.

Here's another pattern that I see, kind of a subversive one. There seem to be two kinds of XML-based initiatives out there. In the first, they don't bother too much with the niceties, they just focus on getting some things put together and happening, they make up and refine the messages as they go along, and they've been in production now for six months. The second kind takes a more carefully structured approach, builds things from the schemas out, worries a lot about choreography and data modeling and semantics, and is still in the planning stage, with the revised schedule calling for deployment two quarters from now if things go well.

Admittedly, I'm exaggerating a bit; but not all that much.

Meanwhile, the biggest story in the XML world is happening just off the radar of the prognosticators and executives. It's called RSS, and it's a simple format for pumping the content of dynamic information sources around. It was invented for use by the legions of webloggers, but it's mainstream now; I no longer surf to the *New York Times* or the BBC or MSDN, I subscribe to them, and when something

~continued on page 8~



WRITTEN BY
BOB ZUREK

Act now to expand your business capabilities

Today's software industry is moving fast to supply innovative technologies, new standards, and early customer case studies targeted at fulfilling the vision of Web services. But as these products, standards, and customers emerge, it is now time to move beyond using the simple term **Web services** – it really misrepresents what is starting to take shape in today's enterprise, the emergence of the enterprise service-based architecture.

A Changing Landscape

Without a doubt, Web services are real in today's enterprise, primarily through the emergence and subsequent adoption of three important, agreed-upon technologies: SOAP, XML, and WSDL, considered the foundation for enabling Web services. However, if we look deeper inside the world of IT, we find other well-established and emerging technologies like Java, Enterprise JavaBeans, JMS, .NET, and CORBA. These technologies further establish the bedrock for a much more important phenomenon that is emerging – an enterprise service-based architecture that is built upon the foundation technologies of Web

services. Java, Enterprise JavaBeans, CORBA, and JMS, in combination with SOAP, XML, and WSDL, are enabling this new enterprise service-based architecture. It's now time for the industry to stop focusing on describing and defining Web services and move toward describing and defining a service-based architecture for developing and deploying information systems. In addition, the vendor community should now focus on describing how their particular solutions fit into this new architecture and the benefits they deliver to today's enterprise information systems.

Filling in the Gaps

As enterprises begin developing an enterprise service-based architecture, much of their existing legacy code will need to be accessed as services to leverage the underpinnings of this new architecture. Existing code may have been built using Java or Enterprise JavaBeans, or invoked using message-based systems. The challenge is in enabling the chunks of software in a service-based architecture when you have to deal with different protocols or are concerned about the performance implications of

interfacing, for example, your Enterprise JavaBeans through SOAP versus through a native call to the bean.

The good news is that development environments like Visual Studio and BEA WebLogic Workshop allow the rapid construction of Web services for new and existing code. In fact, many vendors are doing quite a bit to expose critical functional components of their solutions as Web services. All you have to do is go to Google and type in the name of your software vendor and the term "Web services," and you will likely get a good list of links taking you to information on their Web services capabilities. In addition, innovators from many areas of the industry have come together to build an infrastructure for invoking services under a promising project that is part of the Apache Software Foundation, WSIF (Web Services Invocation Framework). In a nutshell, WSIF is a Java-based API that allows your client code to invoke WSDL-based service through a layer of abstraction rather than natively through SOAP. The abstractions are the pieces of WSDL that include the port types, operations, and message conversations that don't refer to the actual protocols being used. According to the Apache WSIF site, "the abstract invocations work because they are backed up by protocol-specific pieces of code called providers. A provider is what conducts the actual message exchanges according to the specifics of a particular protocol – for example, the SOAP provider that is packaged with WSIF uses a specific SOAP engine like Axis to do the real work. The decoupling of the abstract invocation from the real provider that does the work results in a flexible programming model that allows dynamic invocation, late binding, and clients being unaware of large-scale changes to services – such as service migration, change of protocols, etc.

"WSIF also allows new providers to be registered dynamically, so you could enhance your client's capability without ever having to recompile its code or redeploy it." Furthermore, according to the WSIF site, "using WSIF WSDL can become the centerpiece of an integration framework for accessing software running on diverse platforms and using widely varying protocols. The only precondition is that you need to describe your software using WSDL, and include in its description a binding that your client's WSIF framework has a provider for. WSIF defines and comes packaged with providers for local Java, EJB, JMS, and JCA protocols. That means you can define an EJB or a JMS-accessible service directly as a WSDL binding and access it transparently using WSIF, using the same API you would for a SOAP service or even a local Java class."

Combine the functionality of WSIF with the fact that many solutions are available for wrapping existing code and exposing it as Web services, and we begin to see that we have a world in which we're able to interact with the services in an abstracted model that is protocol independent. WSIF efforts must now expand on the number of providers. Currently providers exist for JCA, EJB, and JMS. It is hoped that the list of providers will expand over time and that we will see WSIF supported in other enterprise software solutions.

A New Server Emerges to Support Service-Based Architectures

Popularized by the emergence of technologies like Java and .NET – along with the emergence of the enterprise application server from companies like IBM, Microsoft, BEA, Novell, and Oracle – over the years, many enterprises have shifted the focus of their development efforts from a client-server architecture to an application server-centric architecture. Today, however, we are starting to see the emergence of a new class of server platforms that will help the enterprise move from a server-centric

architecture to a service-based architecture; what I call the enterprise service platform. Already, companies like IBM and BEA are rapidly expanding the capabilities of their application servers and surrounding tools in order to support the creation of an enterprise service platform. The beauty of this expansion is that enterprises with existing application servers will be able to leverage their existing solutions, skills, and knowledge, and begin to take advantage of the new capabilities of these enterprise service platforms to move them quickly toward service-based architectures.

Expanding Support for the Service-Based Architecture

Further evidence of the emergence of this new enterprise service platform and service-based architecture is the wide spectrum of independent software vendors, ranging from start-ups to specialists, that are filling the gap in functionality by developing supporting infrastructure to the emerging service platform. Two particularly busy areas for opportunistic entrepreneurs are service-based management and service-based business process management systems.

One area of focus inside today's enterprise is that of streamlining various business processes with the objective of reducing latency in the business in order to respond more rapidly to increasing customer and partner demands and to become more operationally efficient. All this comes with a drive toward achieving further cost reductions and profitability in the business. Combine this with the emergence of industry regulations like HIPPA and Basel, and the topic of optimized business processes is bubbling up to the boardroom in today's enterprises.

This renewed interest in business process optimization will go far in pushing an enterprise toward a serviced-based architecture. In fact, industry analyst firm ZapThink is already calling this "Service-Oriented Process" and claims it will be an \$8.3 billion market by 2008 and will drive 70% of Web services implementations. Although this is a bold claim, service-based architectures will certainly require that a business process manager be available to the enterprise to effectively orchestrate the services in the context of the overall business process. Service-based architectures will go a long way in simplifying how services are orchestrated; in fact, many believe that we will soon see a time when business-level analysts, not IT professionals, will build, streamline, and manage the business processes within an enterprise using these service-oriented process tools.

Further enforcing the direction toward a service-based architecture driven by business processes are enterprise software vendors like SAP, Siebel, and PeopleSoft. Already these vendors are pushing service-based business processes as a way of integrating their systems with other custom or packaged applications. Siebel Systems has invested substantial dollars in a project they call Universal Application Network (UAN). The objective of UAN is to provide a core set of prebuilt business processes supported by the Business Process Execution Language (BPEL) standard to make it easier to integrate systems through the use of exposed services. The intention of UAN is to enable any business process orchestration vendor that supports BPEL to interoperate and support the UAN business processes. To enable UAN in an enterprise, the enterprise must be in the mode of moving toward an enterprise service-based architecture. The same goes for SAP and PeopleSoft. Again, as the enterprise application vendors ramp up on service-based architecture, it will be easier for enterprises to adopt components of the enterprise applications as services in the context of their overall business processes.

Once an enterprise embarks on a mission to move toward a service-based architecture, they will need to adopt a service-based management framework. Again, this is a hot spot in the area of emerging service-based solutions. The purpose of the management framework is to help the enterprise observe, organize, report, and act on the services deployed inside the enterprise, regardless of where they originated, how they are invoked, or what other services they depend on. Emerging companies like Adjoin, Actional, and Confluent, and even larger established enterprise players like HP and IBM, are all building out service-based management tools and technologies to help enterprises get control of the runtime characteristics of their services. These solutions vary in their capabilities and ability to reside inside the enterprise service-based architecture, but will certainly be part of the overall software stack needed to ensure the success of a service-based runtime environment.

Open Source: A Maturing Player in Enabling a Service-Based Architecture

In today's highly competitive market, there are hundreds, if not thousands, of available commercial and proprietary software vendors and system integrators that will support an enterprise initiative to move towards a service-based architecture. In addition to these commercial offerings, there is a huge wave of open source initiatives, some highly visible like Linux and JBoss, and some just getting started. The community and collaborative efforts amongst the open source efforts are powerful and moving fast and furiously to support alternative solutions to proprietary commercial service-based offerings. This certainly has the commercial software industry watching very closely. However, even commercial software vendors are getting into the game. Take IBM, for example, who is carefully straddling the fence between open source and commercial efforts around enabling the creation of a service-based architecture. IBM played a key role in the efforts around WSIF, as mentioned earlier in this article. With the rapid emergence of open source, enterprises would do well to start preparing themselves to potentially take advantage of this rich source of new and emerging technologies as they embark on a service-based architecture.

One way to approach this open source topic is to establish a team of skilled professionals whose purpose in the enterprise is to investigate and experiment with open source solutions in order to gauge whether or not the open source solution could be an alternative to a commercial offering in support of a service-based solution. However, before ever deploying an open source solution, an enterprise should also have a person or team of people developing policy and procedures, in conjunction with the company's legal department, for how open source is used, supported, and deployed within the enterprise. In fact, because it is so easy to download open source products along with fragments of open source code, it wouldn't be unusual for an enterprise to have already been injected with open source without the company knowing it.

Having good policy, processes, and practices will go a long way to control the use of open source without getting heavy handed with your developers. The licensing and use of open source is all over the board. Without this level of expertise, enterprises may put themselves at risk by not having appropriate policies.


Begin Your Initiative Now, Your Competitors Are

With major enterprise software vendors and open source efforts all pointing toward support of the emergence of enterprise service-based architecture, companies must start their efforts now or face increased competition from enterprises that jumped on the early adopter bandwagon and experienced success. I frequently like to use Kinko's as an example of a company that saw the opportunity to leverage a service-based approach to expand their business model.

Kinko's recently announced that they will be exposing a printing service as a Web service to Microsoft Word users. With the Kinko's service enabled, it is envisioned that Word users, choosing the print command, will see the standard print dialog but, in addition to seeing all the printers, they'll also see a "Print at Kinko's" option. Selecting the service will allow users to pick the Kinko's closest to their physical location in order to have the job printed at the chosen Kinko's location. I'm sure that a message will travel back to the user indicating that the print job is complete and ready to be picked up. Who knows, maybe even a map from MapQuest will be displayed showing the exact location of the Kinko's. Imagine how, by taking this approach, Kinko's would be able to expand their business model by making it very convenient for their customers to take advantage of this future service.

Expanding the concept further, imagine Kinko's interacting with a Federal Express service where the user would simply send a list of mail addresses to the Kinko's service, requesting that the final print jobs be sent to Federal Express for delivery to the specific addresses. This is quite visionary on behalf of Kinko's. With success and good user adoption, this service might serve as a "poster child" to demonstrate how, by taking a service-based approach to their business, Kinko's is able to better serve their customers and further expand their business visibility.

...

Today we are in the early cycle of this new service-based architecture. In the future, we will see the benefits to businesses who, like Kinko's, saw the opportunity to move to a service-based offering and expanded the capabilities of their business. 

AUTHOR BIO


Bob Zurek is responsible for Ascential's overall product strategy, including new product development and technology acquisition. With more than 23 years of high-tech experience, Bob is instrumental in developing and driving Ascential's enterprise integration strategy, including the company's parallel processing framework, data quality, and Web services strategies.

BOB.ZUREK@ASCENTIALSOFTWARE.COM

GUEST EDITORIAL

~continued from page 5~

changes, I get a nice little summary and decide whether I want to check it out.

RSS has never actually been blessed as a standard, and its development has been fraught with nasty personalities and politics. There are competing versions, and the next-generation version probably won't be called RSS. But it's changing the world, and it's based on XML, and it's coming from a direction that nobody's looking in. Stand by. 

AUTHOR BIO

Tim Bray founded Antarctica Systems, a pioneer developer of data visualization technology. In 1989, Tim cofounded Open Text Corporation, where he developed high-performance text retrieval software and in 1994 introduced what would become one of the first commercial Web search engines. In 1998, he co-invented Extensible Markup Language (XML). More recently, Tim was named to the InfoWorld Hall of Fame and Upside's Elite 100.

TWBRAY@ANTARCTICA.NET

Isavix

www.isavix.net

Introducing Microsoft InfoPath 2003

Part 2

WRITTEN BY
THOM ROBBINS

Extending InfoPath solutions with BizTalk Server 2004

In Part 1 of this article (*XML-J* Vol. 4, issue 6) we looked at creating a solution that used a new product in the Microsoft Office System 2003 called InfoPath. In this installment, I'll show you how to extend the solution created in Part 1 using BizTalk Server 2004.

In Part 1, we created a form that collected new patient information for a health care customer. Once we completed the initial InfoPath solution, the CIO was very impressed with the fact that InfoPath leveraged existing back-end Web services and allowed the company to continue developing using the service-oriented architecture (SOA) that they had invested very heavily in. The CIO's next challenge for my project team was to build an InfoPath solution that extended the patient form and included workflow components. What he wanted was an easy integration into their back-end mainframe system. I can remember his exact words, "Integrate and they will come!" Now we'll do exactly that. I will show you how we extended the solution using BizTalk Server 2004 and how the enhancements in this version make it possible.

What Is BizTalk Server 2004?

BizTalk Server 2004 is the third release of this server product. When we started this project BizTalk Server 2004 was publicly available in Beta 1 (www.microsoft.com/biztalk/beta). This new version is designed to solve three basic common integration scenarios – Enterprise Application Integration (EAI), Business Process Automation (BPA), and Information Worker Integration. The redesigned architecture is built on two main services that enable these: orchestrations that provide the execution framework for business processes, and BizTalk Messaging that enables transformation and routing between business processes.

Orchestrations allow the design, execution, and management of a business process. Workflows are created and saved into executable XML files called an XLANG orchestration.

XLANG is a proprietary execution environment of BizTalk Server, but the new version enables the exporting of orchestrations to the Business Process Execution Language for Web Services 1.1 (BPEL4WS) specification. BPEL4WS defines a language for the formal specification of business process and business interaction models. XLANG has been a core component of BizTalk since the first product release but has been substantially enhanced within the new version. XLANG is now an extension of the managed runtime environment of the .NET Framework and this inclusion provides enhanced support for communication, transactions, long-running processes, and state management. These new managed orchestrations appear as a palette in Visual Studio .NET 2003 that allows developers to visually create, edit, and modify workflow processes directly within the development environment. For example, the managed environment enhancements and the

“Orchestrations allow the design, execution, and management of a business process”

new BizTalk Architecture allowed us to develop our initial NewPatientOrchestration without any custom coding. An additional extension to XLANG is a tracking engine. The Business Activity Monitor (BAM) allows business decision makers to monitor business processes and track potential process bottlenecks. Using the dashboard interface of Windows SharePoint Services the business user is able to monitor, interact with, and even change existing BizTalk processes. The main feature of BAM is the provision of a platform for continual process improvement that is easily configured and managed.

Within BizTalk Server 2004 both messaging and orchestra-

tion have been combined into a single subsystem. This combination enables a shared common data store for process, state management, and messaging. During the execution of a process the messaging components handle the transport and mapping while the orchestration executes the defined processes.

The BizTalk Implementation

The CIO tasked the project team with two goals for the NewPatientOrchestration. The first was to publish a Web service that accepted the incoming completed and validated InfoPath forms. The second was to implement a basic transformation that could easily match an existing mainframe upload XML Schema (see 'Defining the Architecture').

The direct integration of BizTalk into the Visual Studio .NET 2003 IDE adds a new set of project types. For the NewPatientOrchestration and Web service we started with an empty BizTalk project. This project type created a blank project with references to the managed assemblies needed to create an orchestration. The inclusion of BizTalk into the managed environment means that all default references exist as a managed assembly. These default references include:

- Microsoft.BizTalk.DefaultPipelines
- Microsoft.BizTalk.GlobalPropertySchemas
- System
- System.Xml

One of these references is to a new feature called pipelines. These define a .NET or COM component that defines and links to the various processing stages of either sending or receiving a message. Each portion of these components specifies a messaging stage. For example, these stages include encoding/decoding, assembly/disassembly, and encryption/decryption. The pipeline designer allows the developer to modify, change, or augment the specific order of these stages.

The Microsoft.BizTalk.DefaultPipelines namespace defines the default pipelines: XMLReceive, PassThruReceive, XMLTransmit, and PassThruTransmit. This default reference processes documents using the PassThruReceive and

Pipelines

A pipeline is a new feature of BizTalk Server 2004 and defines a .NET or COM component that implements a set of predefined interfaces that interact with the BizTalk engine. The default order of the pipeline process for each of the components defines the default set of interfaces and processing order.

Default component order in the XMLReceive pipeline:

- Decrypter
- Decoder
- Disassembler
- Validator
- Party Resolution

Default component order in the XMLTransmit pipeline:

- Assembler
- Encoder
- Encrypter

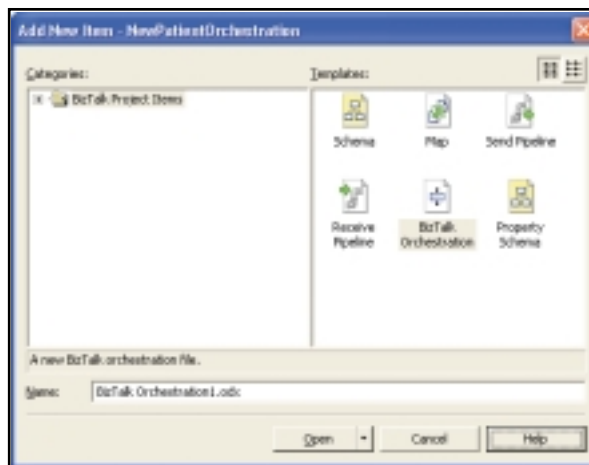


Figure 1 • XSDs for orchestration

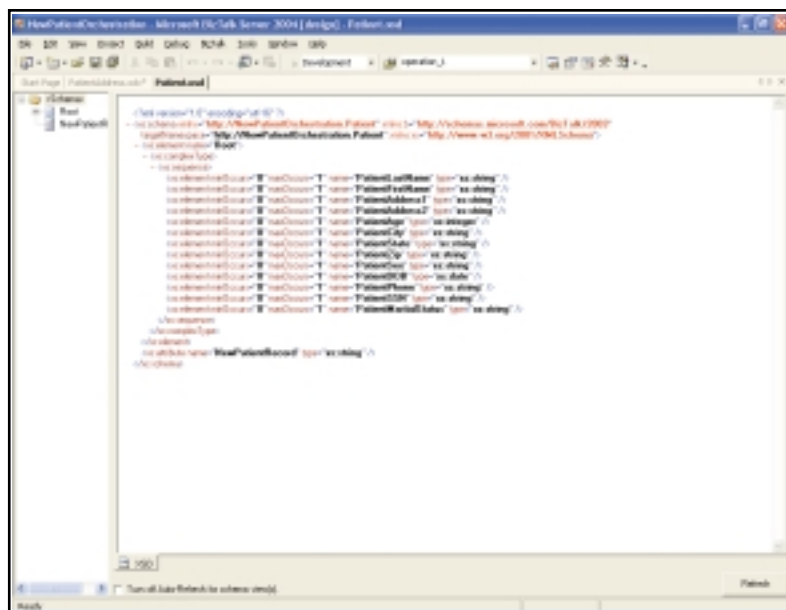


Figure 2 • Inbound XML document

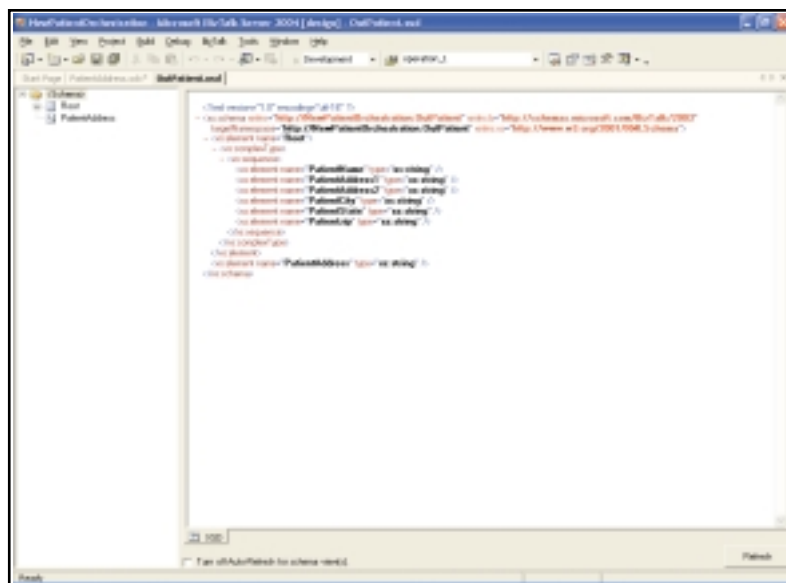


Figure 3 • Transformation

PassThruTransmit function (see sidebar ‘Pipelines’).

Defining Messages

Pipelines are used to process messages that are defined as an XML Schema (XSD). These schema definitions define and maintain the specific message formats and structures used during the orchestration. We created two XSDs that matched the specific schema items and formats that were needed for our orchestration. These were generated using the new XSD schema item within the Visual Studio 2003 IDE (see Figure 1). The first schema (see Figure 2) represented the InfoPath-submitted inbound XML document. The second schema represented the transformation and matched the existing XML message format used for the mainframe system (see Figure 3).

Once we completed both the inbound and transformed schemas it was important to check the specific XSD formats. In many ways, reviewing and comparing a specific message instance is a lot easier than reviewing the schema tree and data types. Creating an XSD can be a very complex process that includes describing all the possible formats and variations within a specific document type. Within the Solutions Explorer, right-clicking the schema and selecting “generate instance” creates a reference document for schema definitions. It is important to remember that this is not a foolproof check, but it really helped to quickly review and validate the XML document.

Orchestration Design

Messages are the basic unit of communication within an orchestration. Messages consist of one or more parts with context data that describes their properties and content. BizTalk orchestration uses messages to provide the context for interchange between business process participants. Typically, orchestration is broader in scope than a traditional workflow. Each participant is autonomous, and the responsibility of routing a work item is determined by each cooperating participant. BizTalk Orchestration extends the definition of the traditional interaction diagram to include definition and control for decisions, concurrent actions, transactions, and supporting actions. The result of a completed orchestration is an XLANG executable file that represents the description of the business processes.

Within the Visual Studio 2003 IDE, new orchestrations are added in the same way as all other BizTalk items – through the Solutions Explorer and the New Item menu. This adds a file with an ODX extension that contains the XLANG structures.

Within the IDE the orchestration page provides a visual interface that presents the underlying XLANG structure. This interface is broken into three sections that define an orchestration. First is the actual design surface or palette. This area is the visual surface where toolbox items are dropped and configured to define the process flow. Using this surface I dropped the send and receive ports that define the inbound and out-

“...reviewing and comparing a specific message instance is a lot easier than reviewing the schema tree and data types”

bound message handler. The second area of the orchestration designer is the Orchestration Types window, which defines the specific port types, correlation types, and role links available within the orchestration. Using this window I was able to define the specific inbound and outbound operations defined for the orchestration. Finally, there is the Orchestration Variable window, which shows the orchestration properties, parameters, ports, messages, variables, correlation sets, and role links available for this orchestration. Using this window I was able to correlate the specific message to either the inbound or outbound port (see Figure 4).

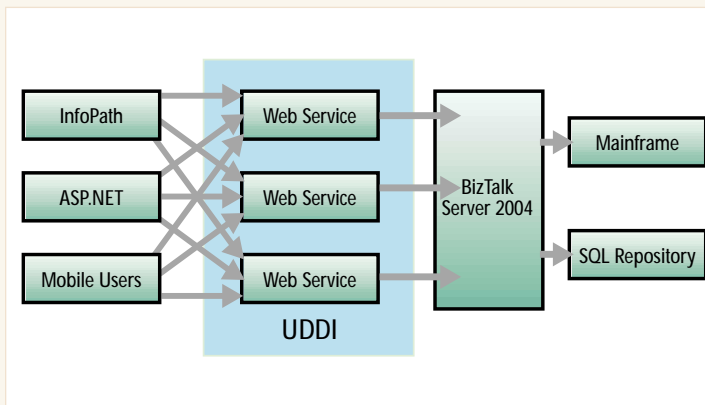
Mapping schema definitions

Within the orchestration designer, the transform shape enables the flow of data from one message to another. This can be done through a direct assignment or a mapping process. The NewPatientOrchestration required a specific map because of required transformation and data translation to the outbound message type. The transform shape provides access to the BizTalk Mapper, which allows the assignment of specific source and destination attributes and provides both transformation and translation of specific schema elements and attributes.

Maps are initialized and defined by placing the transform shape onto the design palette (see Figure 5) and then defining the schema for the source and destination documents. This configuration is then loaded into the BizTalk Mapper, where the actual element and attribute flow from one message to another is defined. Within a map, functoids are used to manip-

Defining the Architecture

Extending InfoPath Solutions to include BizTalk orchestrations using BizTalk Server 2004 is even easier. BizTalk now offers many additional features including direct Web service publishing and easy integration for the extension of Web services. Within the reference architecture BizTalk becomes a set of Web services that provide workflow and orchestration services.



ulate the data flow. Functoids are an important part of any map. They provide the ability to map elements and attribute data into different elements and attributes across different structures. Within BizTalk Server 2004 they have become part of the Visual Studio toolbox palette and appear as part of the map. When working on a map you can drag them directly onto the design palette. For example, within the NewPatientOrchestration we dragged the string concatenation onto the map and dragged the source and destination elements between the two (see Figure 6). Within a map, input links correspond to input parameters and lead to the functoid from the left; an output link corresponds to the output parameters and leaves the functoid from the right.

Deploying the Solution

Once the map and orchestration are completed, the solution is ready for deployment. All projects within BizTalk Server 2004 are deployed into a managed assembly as a DLL. This is done by building the solution and then deploying it using the E-Business Web Services Wizard. This wizard provides the option of deploying either an orchestration or an enterprise schema as a Web service. When deploying an orchestration, the wizard starts with the compiled DLL and generates the necessary namespace, Web service name, communication patterns, and request/response objects to build and deploy an ASP.NET Web service project. The wizard identifies the defined ports, location, and schema within the compiled assembly to generate the necessary WSDL file that provides the Web service definition.

Any orchestration that uses only a single Web port is published as a single ASMX file that defaults to the orchestration name. The orchestration that we created used more than one Web port, and the default naming context is the name of the orchestration with an underscore followed by the Web port name. For example, once the NewPatientOrchestration was deployed, the Web port names NewPatientOrchestration_Outbound.asmx and NewPatientOrchestration_Inbound.asmx were assigned. It is important to keep these in mind within InfoPath as these are the service names that that InfoPath will bind to. Once bound to the specific Web service InfoPath has the ability to directly submit and receive data from these defined BizTalk Services.

Summary

In this article we developed a fully functional BizTalk process that was deployed as a Web service. We have covered many of the features of BizTalk Orchestration and deployment within the managed environment of the .NET Framework. The real benefit of this architecture was the creation of a Web service that offered workflow and integration capabilities that were easily accessible within the InfoPath environment. Throughout these two articles we have only covered the basics of InfoPath and BizTalk Server. There are still many more features that my health care customer found compelling. I challenge you to take the time to explore these and see the power of InfoPath to help reduce the mountains of paper that we have and BizTalk Server 2004 as a way of providing workflow and structure around existing business process.

AUTHOR BIO

Thom Robbins is a senior technology specialist with Microsoft in New England. He spends his time working with customers on developing and implementing solutions with Microsoft-based technologies. He is a regular speaker at various industry events and a frequent contributor to various trade magazines.

TROBBINS@MICROSOFT.COM

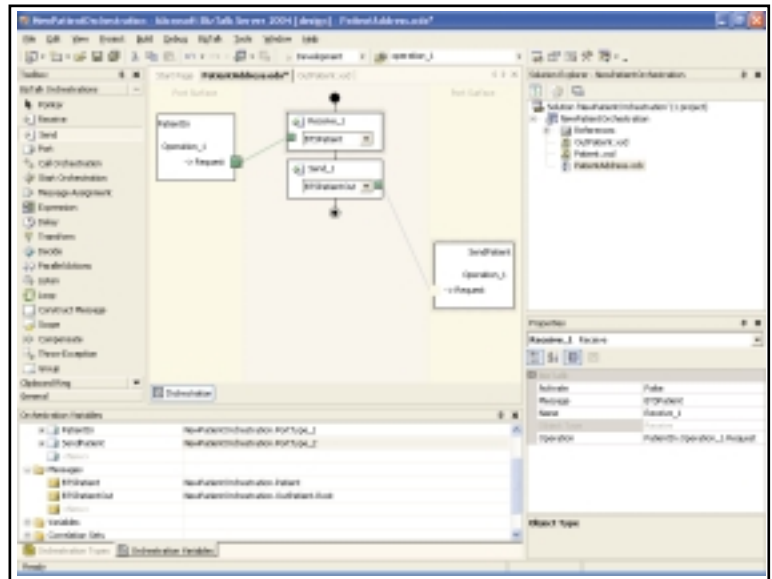


Figure 4 • Orchestration Variable window

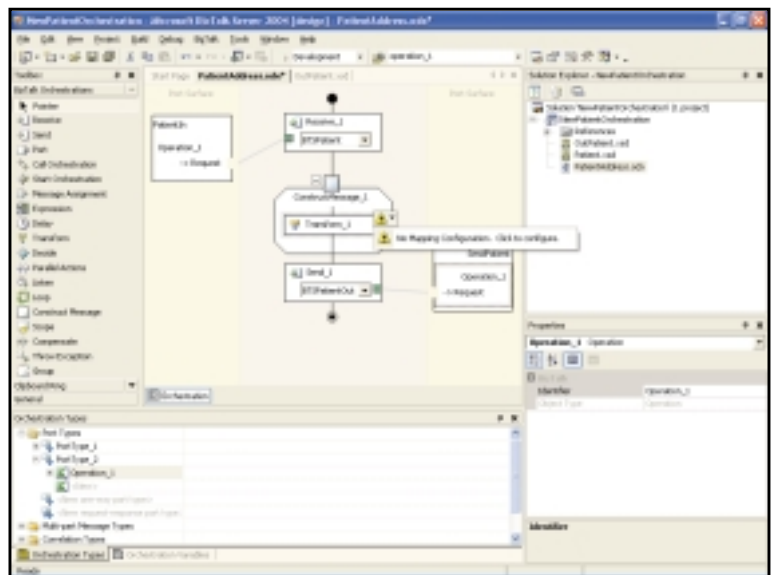


Figure 5 • Placing a transform shape on the design palette

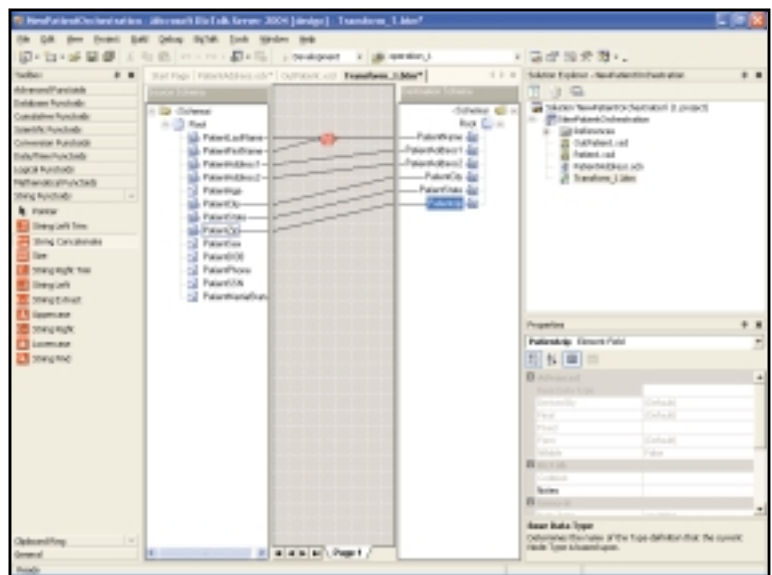


Figure 6 • Function dragged onto design palette



Aggregation with XMLSPY

The advantages of XML...without all the angle brackets

Aggregation in XML is not trivial. Altova's XMLSPY offers a number of features facilitating this process. This article presents an example, including best practices and practical programming techniques – especially useful for those who don't like typing a lot of angle brackets.

Aggregation of XML (or HTML) documents means to collect the content of several XML files in one XML (or HTML) document (see Figure 1).

A portal product, for example, would aggregate the content of several data sources into one HTML page and present their contents in boxes in the user's browser. Most portals do this with programs written in Java, Perl, or some other programming language; however, XSLT includes the function `document()`, which is suitable for this purpose.

XMLSPY is a high-level XML editing tool, offering many visual capabilities for creating, changing, formatting, and presenting XML documents. The following example shows how XMLSPY's features can be used for aggregation.

AUTHOR BIO

Andreas Solymosi is a mathematician and graduated from the Leningrad State University, Soviet Union. He earned a PhD in computer science at the Erlangen University, Germany. Andreas teaches programming languages and software engineering at the Technical University for Applied Sciences, Berlin, and is working on an XML portal project in Orlando, Florida.

XMLSPY Features

XMLSPY uses the following file name extensions:

- **.xml**: XML data
- **.xsd**: XML Schema Definition
- **.xsl**: Extensible Stylesheet Language
- **.xslt**: XSL Transformations
- **.sps**: Stylesheet Designer's internal format

An .xml document contains the data. XMLSPY can check (with the function key F7) if it is well formed, i.e., if it satisfies XML's syntax. With its data structure defined in an XSD document, XMLSPY

can check (with F8) if it is valid (if it uses only the structures defined in the schema). The reference to XSD can be written into the XML document.

XMLSPY can automatically generate XSD for an existing XML document; however, it must be reviewed because it might contain undesired constraints.

An XSL document usually contains formatting information for XML data. It can be visually developed with Stylesheet Designer for an existing XSD document, and formatting can be assigned to every structure element contained there. The result can be viewed in HTML preview if a working XML file (with data) has been assigned. Stylesheet Designer stores the result in its internal .sps format (a special XML language) for further processing by XMLSPY's Stylesheet Designer for the "authentic view." Stylesheet Designer can also generate an XSL (or XSLT) document, which can be used by any XML processor (e.g., XMLSPY or an XML-enabled browser such as Netscape 6 or Internet Explorer 6.0). For this the XML document needs to contain a reference to the XSL document or vice versa (see Figure 2).

XMLSPY can process (with F10, with F11 even debug) the XSL document either way. Similarly, a link to SPS can be put into XML (used only by XMLSPY for the "authentic view").

Stylesheet Designer also allows you to take an HTML document, but not an XSD, for the design's base. It can also be saved as .sps and .xsl.

XMLSPY can present XML data in the following views:

- Text view (raw XML, editable)
- Browser view (uneditable), with or without a stylesheet reference
- Enhanced grid view (the best choice for initial data entry)

- Authentic view (editable, the best choice for additional data entry)

For the authentic view it is necessary to have defined an SPS document with Stylesheet Designer. XMLSPY will then present the XML data in the defined format, offering very convenient editing and extending. However, only data elements that exist in the original XML file can be changed or extended. So the steps to working with XMLSPY are:

1. Create sample XML data in XMLSPY's enhanced grid (or text) view.
2. Generate (and review) XSD reflecting the document's structure.
3. Visually create a design (on the basis of XSD) with Stylesheet Designer as SPS.
4. Generate XSL from SPS with Stylesheet Designer.
5. Connect XML with XSD (for validation) and SPS (for the authentic view) in XMLSPY.
6. Edit visually (and create more) XML data in the authentic view.
7. Generate HTML for presentation in any browser, or connect XML with XSL for XML-enabled browsers.

Alternative designs (e.g., through modifying the SPS file) would present the same data in a different style.

Aggregation

The steps presented here solve most of the simple problems in working with XML files. The problem of aggregation, however, is more complex. Here an XSLT document containing a program that completes the aggregation by calling the standard XSLT function `document()` must be developed. An XSLT program consists of a set of rules that can be

applied to parts of the input XML document. The rules define what (text) data should appear in the output document.

There is no principal difference between XSL and XSLT: both document types are processed against an XML document. However, it's a good convention to put formatting information into XSL (as a stylesheet) and transforming information into XSLT documents. XSLT should output XML data, and XSL can output XML or HTML.

The XSLT document `aggregate.xslt` performs the actual aggregation (see Listing 1). `aggregate.xslt` contains two rules: the first one, `xsl:template match`, matches the whole input XML document ("`/`"), generates a `<result>` tag (with an XSD reference), and then looks for `<families>` tags in the XML document (`xsl:apply-templates select`). The second rule matches all the `<location>` tags in the input XML document and copies the document with the URL given in the tag's data.

An example with families shows how this aggregation works – the idea is to have several XML documents describing families (with country, last name, father, mother, and children):

```
<!-- orlando.xml -->
<family>
  <country>USA</country>
  <lastname>Solymosi</lastname>
  <father>Andrew</father>
  <mother>Ingrid</mother>
  <children>
    <child>Esther</child>
    <child>Judith</child>
    <child>Thomas</child>
    <child>Philip</child>
  </children>
</family>
```

and one index file listing all the family documents (named by the residences of the author's family members, suggesting that those XML files can be scattered all around the world, just like today's families):

```
<!-- index.xml -->
<families>
  <location>orlando.xml</location>
  <location>erlangen.xml</location>
  <location>polling.xml</location>
  <location>budapest.xml</location>
</families>
```

The goal of the aggregation is to present all the families on one page.

Table 1 shows the result of aggregating the files and presenting their content in a table with a stylesheet. There are two

Country	Last name	Father	Mother	Children
USA	Solymosi	Andrew	Ingrid	Esther Judith Thomas Philip
Germany	Pröll	Johannes	Ursula	Ingrid Liselotte Dieter
Germany	Solymosi	Peter	Rosmarie	Johannes Mathias
Hungary	Solymosi	Sándor	Vera	Andrew Peter

Table 1 • All families presented on one page

main steps in this process: aggregation and presentation. They are described in the two files `aggregation.xslt` and `style.xml`. In `aggregation.xslt` we program the aggregation (i.e., pulling the content of the documents together); in `style.xml` we store formatting information.

All these files can be downloaded from www.solymosi.com/Andreas/Family/Aggregation.html.

Example: Aggregation with XMLSPY Create documents to aggregate

In this section we're going to create XML documents with data to aggregate. Our goal is to use XMLSPY's features and work as little as possible on the XML level (and to avoid typing `<angle brackets>`!).

1. Create the sample data file `orlando.xml` (it does not have to list all four children, just the first two).
2. Generate schema file `family.xsd` (XMLSPY's Menu: DTD/Schema, Generate DTD/Schema, W3C Schema, OK, `family.xsd`, edit it, and delete constraints for the children's names – otherwise no more children can be added).
3. Open `family.xsd` with the Stylesheet Designer.
4. Design and save `family.sps` (test it with data in `orlando.xml`).
5. Generate and save stylesheet `family.xml`.
6. Open the data file `orlando.xml` with XMLSPY.
7. Connect it with the stylesheet `family.xml` (for the browser view) and with `family.sps` (for the authentic view).
8. Add the rest of the children in the authentic view, correct misspelled names, etc.
9. `orlando.xml` can now also be viewed in any XML-enabled browser because it contains a stylesheet reference (see Figure 3).

The document `family.xml` is not necessary if the data documents are not going to be presented in a browser. fam-

ily.sps is necessary only if they are going to be visually edited in XMLSPY (a very convenient feature). `family.xsd` is necessary every time an XML processor is going to validate a data document (like XMLSPY does when opening it).

Create aggregation file

In a similar way we can now create our aggregation file containing the information about which documents will be aggregated. It is a kind of "table of contents" and is going to be the starting point of the aggregation. This is why it's called `index.xml`:

10. Create the sample index file `index.xml` (as before, but with two location tags).
11. Create the metadata documents (`index.xsd`, `index.sps`, perhaps `index.xml`) as before.
12. Add the rest of the data (list of documents to be aggregated) in the authentic view.

Note: In step 10 we created the index file with two location tags; not with four (or more) because in the authentic view (step 12) it's easier to edit; not with one so that the schema file `index.xsd` con-

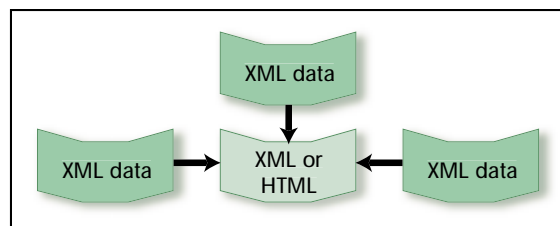


Figure 1 • Aggregation of XML documents

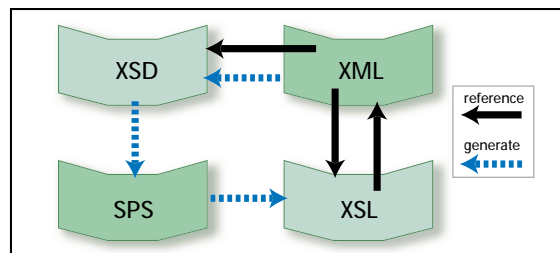


Figure 2 • Reference to XSL document

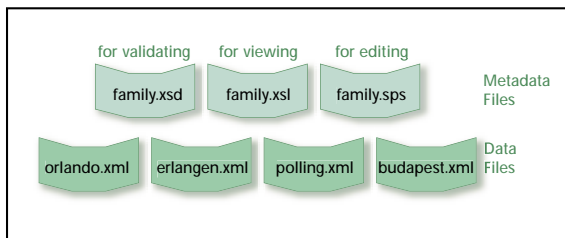


Figure 3 • Stylesheet reference

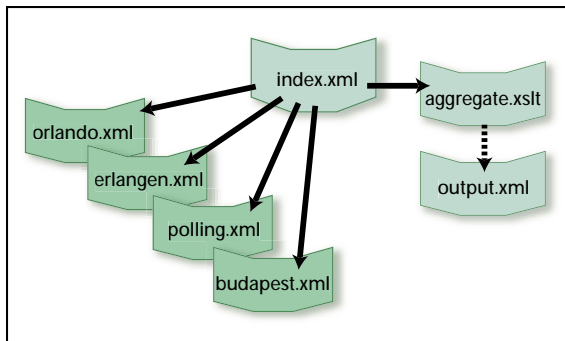


Figure 4 • index.xml

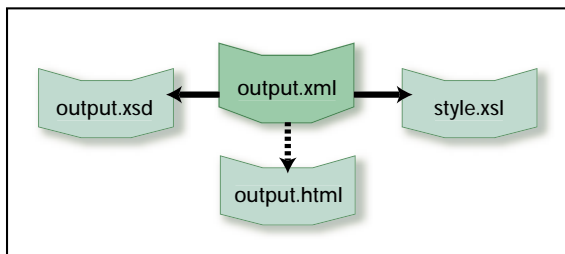


Figure 5 • Authentic view

tains the repetition. If the complete index file is created in step 10, steps 11–12 can be omitted.

Program aggregation

Now aggregation can be programmed and executed:

13. Write `aggregate.xslt` (as in listing 1, which can be found at www.sys-con.com/xml/sourcecfm) – this is the only step XMLSPY doesn't offer great support for.
14. Set in XMLSPY's menu: Tools, Options, XSL, Default file extension for output file: `.xml`.
15. Connect in XMLSPY `index.xml` with `aggregate.xslt` (menu XSL, Assign XSL, OK, Browse, `aggregate.xslt`).
16. Run aggregation with F10 or with menu XSL/Transformation.
17. Save `output.xml`.

The aggregation process (step 16) can also be debugged with Alt-F11 and F11. The aggregation can be started either with `index.xml` (as described) or with `aggregate.xslt`. In this case a working

XML file must be assigned (menu XSL, assign sample XML). The assignment

```
<?xmlspysamplexml index.xml?>
```

can be written into the document `aggregate.xslt` (after the `?xml` instruction) – this is a processing instruction evaluated by XMLSPY. Alternatively, the document `index.xml` can contain the reference

```
<?xml-stylesheet type="text/xsl"
href="aggregate.xslt"?>
```

in order to eliminate step 15 (see Figure 4).

Design presentation

Now we need a stylesheet for the aggregated document. The hardest part is learning how to handle Stylesheet Designer with more complex schema documents (step 20) – but there's a good tutorial.

18. Generate the schema document `output.xsd`.
19. Open `output.xsd` with Stylesheet Designer.
20. Design and create `style.sps` (test it with `output.xml`).
21. Generate the stylesheet document `style.xml`.
22. Assign `style.xml` to `output.xml` with XMLSPY's menu XSL, Assign XSL, `style.xml`.
23. Now `output.xml` can be seen in browser view.
24. Save `output.xml` and open it in any XML-enabled browser.
25. XMLSPY's menu Tools, Options, XSL, Default file extension for output file: `.html`.
26. Process `output.xml` with F10, save `output.html`, open it in any browser.
27. Assign `style.sps` to `output.xml` with XMLSPY's menu Authentic, Assign configuration file, `style.sps`.
28. Now `output.xml` (not the original family files!) can be edited in authentic view (see Figure 5).

Step 22 (assigning `style.xml`) can be eliminated if the assignment is generated by `aggregate.xslt`; it should then contain the following instruction:

```
<xsl:processing-instruction
name="xml-stylesheet"
type="text/xsl" href="style.xml"
/>
```

before the line

```
<result>
```

Additional Considerations

XMLSPY's XSLT processor (like that of any browser) doesn't allow XSLT pipelining, i.e., it is not able to process more than one XSL or XSLT document at one time. This is why we had to save `output.xml` (after aggregation) and complete formatting in a second step (either in the browser or in producing `output.html`). Some other XSLT processors (like Cocoon or AxKit) follow W3C recommendations and process XSLT documents step by step. So `index.xml` may contain references to more than one XSL/XSLT (and also SPS and XSD) document.

```
<!-- index.xml -->
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="aggregate.xslt"?>
<?xml-stylesheet type="text/xsl"
href="style.xml"?>
<?xmlspysps index.sps?>
<families xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="
index.xsd">
```

XMLSPY and popular XML-enabled browsers (like Netscape 6 or Internet Explorer 6.0) would perform only the first step (`aggregate.xslt`) and present the result without formatting by the stylesheet. This is because they are not designed for actual XSLT processing, just for formatting with a stylesheet. Many XSLT programmers might tend to solve the problem by combining `aggregate.xslt` and `style.xml` into one stylesheet document. However, I believe it's better to separate transforming XML information (which is structural) from formatting HTML information (which is presentational).

Maybe the greatest advantage of using XMLSPY for those who don't like angle brackets (i.e., low-level XML editing) is its authentic view for visual editing of XML data. Altaova also offers a plug-in for popular browsers, allowing XML editing in Web clients (without XMLSPY installed).

Conclusion

Though XMLSPY isn't intended to be an XML programming tool, its features can be very useful in preparing and testing XSL and XSLT documents. Aggregation of XML files is an interesting example where a number of techniques and best practices can be introduced. ☎

ANDREAS@SOLYMOSEI.COM

SpeechTech 2003

www.speechtek.com

Hands-on XForms

Simplifying the creation and management of XML information

Organizations have evolved a variety of systems to deal with the increasing levels of information they must regularly process to remain competitive. Business Process Management (BPM) systems presently take a wide variety of shapes, often including large amounts of ad hoc scripting and one-off implementations of business rules. Such systems tend to be developed incrementally, and pose a significant obstacle to continued development and maintenance.

A World Wide Web Consortium (W3C) specification called XForms aims to change this situation. This article compares XForms to ad hoc solutions to produce a real-life application: the creation of XML purchase orders.

Note: The material in this article is adapted with permission from a chapter of the book *XForms Essentials*, to be published by O'Reilly & Associates in August 2003.

AUTHOR BIO

Micah Dubinko is a senior software engineer in Phoenix, Arizona, working for Cardiff Software in San Diego. He has an extensive cross-platform development background specializing in electronic forms systems. Micah is an editor of the W3C XForms specification, and has written numerous articles as well as the O'Reilly book *XForms Essentials*. He can frequently be found speaking at technical conferences and participating on various mailing lists.

The Problem

Of the several efforts that are under way to define XML vocabularies for business, the most promising seems to be UBL, the Universal Business Language. At the expense of being slightly verbose, the vocabularies defined by UBL do a remarkable job of capturing all of the minor variations that occur in real-world business documents across diverse organizations. For the sample application in my book, I chose a purchase order, since that kind of document is well-understood and demonstrates several features, such as repeating line items, that are customarily difficult to implement. This latest version of this solution is available online at <http://dubinko.info/writing/xforms/ubl>.

A template UBL purchase order document is shown in Listing 1 (listings can be found at www.sys-con.com/xml/sourcec.cfm).

The Solution

Any number of technologies could be called upon to implement an application that produces purchase orders based on templates like the one in Listing 1. Prior to XForms, a typical solution would employ HTML forms and JavaScript. Microsoft InfoPath, currently in beta as part of Office System 2003, offers a better user experience than HTML forms, but still relies heavily on scripting through an event-driven model. As the remainder of this article will show, a declarative approach as used in XForms can eliminate a substantial amount of complexity from the overall solution.

Since XForms is designed to be used in concert with a "host language," I chose a combination of XHTML 1.1 and XForms for the solution, even though a DTD for the combined language isn't available. Comments in the online version include additional information on some of the XML specifics that result from the integration. Listing 2 shows the starting section of XForms code in the document head.

The `xforms:model` element is the container for the entire XForms Model, the name given to the definition of what the form does without any regard for how it looks. It is possible to associate an XML Schema with a form, though that option is commented out in this solution, since XML Schema processing would add significant overhead, and the few places that require additional data type information can be easily specified separately. The `xforms:instance` ele-

ment, with an `src` attribute, points to the initial instance data template that was listed earlier. The `xforms:submission` element indicates that activating submit on this form will write XML to the local file system.

Listing 3 continues with `xforms:bind` elements, each of which selects a specific portion of the instance data, applying XForms properties to the selection. Such properties define overall constraints that are in force at all times. In contrast, script-based approaches require separate but tightly coupled code to handle initialization and each entry point where important values can change.

The language used to select pieces of XML called nodes is XPath 1.0, also known as the selection language underlying XSLT, XPointer, and XML Signature. XPath expressions walk a path from a given starting point through an XML document using a syntax similar to file system navigation. XForms includes defaulting rules that simplify most of the XPath selection expressions that point into the XForms model, called model binding expressions. The first model binding expression in Listing 3 selects the one-and-only `u:IssueDate` instance data node, marking it as required and giving it the XML Schema data type `xs:date`, which provides the hint that this particular data should be entered with a date-optimized form control. The second model binding expression in Listing 3 applies to however many `u:Quantity` elements happen to exist at any given time, a quantity which will vary as line items are added or removed, marks all of them as requiring user entry, and gives the XML Schema data type `xs:nonNegativeInteger`. In contrast, a purely script-driven

approach would need to actively watch for the addition or removal of line items in order to apply the properties to altered nodes.

The next few model binding expressions, in Listing 4, set up the two calculations that are fundamental to a purchase order: the total amount for a line item (price times quantity), and the total for the whole order (sum of all line items).

The calculate attribute holds an XPath expression that gets evaluated whenever needed to determine a new value for the node to which it is attached. Just as file system paths can be based on a current directory, XPaths can be relative to a context node, and can even backtrack toward the root node with the familiar double dot (..) abbreviation. XPath expressions can also contain a handful of arithmetic operators and common functions. The calculation for line items is `../u:Quantity * ../u:Item/u:BasePrice/u:PriceAmount`, where the asterisk means multiply, and the operands on either side of it are path expressions, relative to whichever `u:LineExtensionAmount` element is being recalculated. In turn, the calculation for the grand total is `sum(../u:OrderLine/u:LineExtensionAmount)`, which uses the function `sum()` to add up all the values from individual `u:LineExtensionAmount` nodes. Like a spreadsheet, recalculations will occur as needed with no special programming effort, and dependencies among calculations will automatically be handled in the correct order so, for instance, individual line items will always be multiplied out before the overall total is summed up.

The online version includes a few more tricks that space does not allow discussion of here. In XForms, it is possible to define secondary XML instances that can be used for temporary storage in a form. The online solution uses this technique to centrally keep track of possible currency values used in the purchase order.

Listing 5 shows content that begins the body section of the XHTML host document. It gives you a good feel for the kind of code that makes up an XForms user interface. Form controls have names like `input`, `select1`, or `output`, which suggest the intent of the control without committing to a specific incarnation like “radio buttons” or “pop-up list”. This level of abstractness ensures that XForms solutions will be workable across different kinds of devices, including phones, PDAs, and

eyes-free browsers. On the other hand, it leaves room for designers to add customization through Cascading Style Sheets (CSS). As of this writing, a W3C Working Draft for “CSS3 Basic User Interface,” which includes properties that will enable designers to achieve the fine level of control they desire, is available at www.w3.org/TR/css3-ui. The XForms engine that renders the form, however, will make reasonable choices for how to render form controls, including helpful optimizations. For example, browser-class XForms engines will recognize that the first `xforms:input` element is attached to a node with a data type of `xs:date`, and will automatically use a calendar date-picker control.

Other features of XForms user interface markup stand out in Listing 5. The controls all use the `ref` attribute to point to which instance data node should store the user-entered value. (Another option, not used here, is to avoid specific paths in the user interface by using ID and IDREFs.) Each form control element, except for `xforms:output`, has a required `xforms:label` child element, which ensures that form controls will have an accessible label directly associated.

The second form control demonstrates more unique features. A hint attribute `appearance="minimal"` suggests that this part of the interface should be given minimal screen real estate when not activated – for example a pop-up list. Another attribute `selection="open"` indicates that the user should be able to enter arbitrary values not on the list. Further, a kind of repetition is going on here; the single `xforms:itemset` element maps to a number of list choices defined elsewhere.

A similar repetition occurs in Listing 6, which shows the `xforms:repeat` element. The element `xforms:repeat` causes a repetition of user-perceived content, once for each node in a given set of nodes – exactly the behavior needed to populate the line items in a purchase order. All the content of `xforms:repeat` is effectively duplicated as many times as there are line items, which can be dynamically added and removed. The first form control on each line item is `xforms:range`, which allows a smoother way to select a value than typing a number, for example a sliding indicator. The range here is from 1 to 9, with the attribute `incremental="true"`, which causes changes to this control to interactively propagate throughout the document. The rest of the repeating form controls

are similar to ones already used in this solution.

Challenges

The two main challenges facing developers deploying XForms solutions today are deciding on a host language and configuring stylesheets for all target browsers. Eventually XHTML 2.0, including XForms as the forms module, will be finalized, providing a known and stable target for browsers to implement and designers to write toward. Until that time, however, a reasonable approach is to use XForms elements within XHTML 1.0 or 1.1, without the luxury of DTD validation.

Getting the stylesheets right is trickier, but getting easier day by day as XForms engines get smarter about CSS support, and as the CSS specifications themselves get more polished. The test suite for XForms 1.0, at www.w3.org/Markup/Forms/Test, contains some fine examples of CSS that work well across all common XForms browsers. One specific hint, already used in this article, is to include an `xforms:group` element wrapper to cause the contents of the group to be formatted as a block section, which works particularly well inside an `xforms:repeat` element.

Conclusion

XForms has made vast strides in 2003, becoming a technology suitable for production use by early adopters. Already, businesses are using XForms to produce real documents similar to the one shown in this article. The combination of an open standard with a wide variety of both free and commercial browsers makes a powerful business case for deploying XForms solutions.

Unlike many other XML standards, XForms has remained small, simple, and true to its roots – addressing only well-known and well-understood problems, and providing a universal means to express solutions to these problems. Part of the appeal of XForms is the reuse of proven technologies, such as XPath, for which developers are more willing to invest the time necessary for learning. XForms can also leverage existing XML infrastructure, including XML Schema and Web services components. By embracing a declarative approach, XForms can simplify the code involved in BPM systems, as well as make future maintenance tasks easier. ☺

MDUBINKO@CARDIFF.COM



Object-Oriented XSLT

WRITTEN BY
PIETRO MICHELUCCI

A new paradigm for content management

What could be better for managing content than separating data from presentation? How about separating data from data? Believe it or not, XSLT can actually be used to allow for different levels of data abstraction. In practical terms, this can reduce the complexity of managing Web content by an order of magnitude and facilitate code reuse. In essence, what I'm talking about here is object-oriented XSLT, or to add to the alphabet soup: OOX (pronounced *ooks*).

Why Use OOX?

Isolating content from presentation was the original purpose of stylesheet languages. In the conventional approach, there is just one data layer (XML) and one presentation layer (HTML), with XSL transformations (XSLT) in between. This two-layer architecture simplifies Web site management by allowing content providers to edit their data without concern for stylistic issues, and, conversely, by permitting graphics designers to set the visual tone without regard for specific content. While the two-layer model has been fruitful, XSL transformations (XSLT) empower us to extend data abstraction through the use of multiple data layers. Toward this end, I have created a general-purpose XSLT that you can easily use to apply multiple serial XSL transformations to an XML data document. But before we delve into the details, let's consider why this might be useful.

Multiple data layers

Suppose you're developing a corporate Web site in which there are three types of documents that you wish to make available to your site visitors: white papers, press releases, and external reviews. Each document is stored as XML data in its own distinct format according to its respective schema, which defines its structure. For example, a press release usually contains a headline, release date, location, and contact information, whereas an external review document might contain a title, author, publication date, and hyperlink.

Despite the structural differences in these documents, you may see the utility in making their content available to site visitors in similar ways. For example, on the home page you might want to display headlines for the five most recent press releases as well as titles for the five most recent white papers. So even though press releases and white papers have different data structures, your goal is to present their header content in the same way.

Furthermore, your site visitors may wish to read a summary before embarking upon a full-length white paper or external review. Therefore, you will probably want to give them the ability to view either a white paper abstract or the synopsis of an

external review. Here again we see the need to use the same presentation format for similar content elements that are embedded in structurally dissimilar data documents.

The old way

Using the canonical, two-layer approach, you would need to create one XSL transformation to convert each data document into each presentation format. In our example, we have discussed three data documents and two presentation formats. Therefore, you would need six (three times two) distinct XSL transformations (see Figure 1).

The new way: OOX

Using data abstraction, however, you can add an intermediate data layer to your architecture. The structure of data objects occurring in this layer might be defined by a "generic" schema, suitable for representing content from all text-based corporate documents. With the addition of this layer, the characteristics that normally distinguish different corporate documents are transparent to the presentation transformations. Each presentation transformation only needs to be aware of the data structure for the new, generic documents. Thus, the schema for the new layer of data objects becomes a single interface for the presentation transformations. With this approach, you would need only five distinct XSL transformations (see Figure 2).

The key idea here is that if you can first transform each data document (e.g., press release, white paper, or external review) into a general-purpose corporate document, then you only have to build your presentation transformations for the more general data structure. Furthermore, if you were to add a new type of document in the future, such as "product descriptions," you wouldn't have to create a new XSLT for each presentation format. Instead, you would need only a single new transformation that converts product descriptions into the general-purpose document. This potential reuse of the two existing presentation transformations by leveraging the middle layer schema as an interface illustrates a fundamental advantage of employing an OOX methodology.

Efficiency

In this example, shifting from a two-layer to a three-layer architecture reduced the number of necessary transformations by only one. However, it is easy to imagine the combinatorial explosion that might occur with a more complex architecture. By merely expanding the original specification to include five document types and five presentation formats, the two-layer architecture would call for 25 distinct transformations. In contrast, the three-layer model would only require 10 transforma-

tions. Higher-order abstraction could provide even greater economy.

Flexibility

This three-layer approach, however, is not suitable for all situations. The reason XML data objects conform to different schemas in the first place is that they represent different kinds of information. Press releases adhere to a different structure than external reviews because they consist of intrinsically different content. Insofar as press releases and external reviews are similar, the construction of a general-purpose middle-layer document is useful. But there will certainly be times when it is necessary to present content that is unique to a specific document type. In such cases, the traditional approach of direct transformation applies. Fortunately, in OOX both approaches can coexist peacefully.

Modularity

The advantages of OOX are not limited to just economy of representation. Incorporating multiple layers of data abstraction permits architectural modularity, which leads to ease of code maintenance. One of the key advantages of object-oriented programming (OOP), when implemented properly, is the reduction of dependencies among elements that interact with each other. Due to its modularity, this same advantage applies to OOX. In the context of our corporate documents example, if you decided to change the presentation of the header lists in the traditional model (see Figure 1), you would have to modify each of the three presentation transformations that were created to display those lists. In the OOX model (Figure 2), however, there is only one transformation associated with presenting a list of header content. Therefore, all of the necessary modifications could be made in one place.

How Does It Work?

"OK, I'm convinced that OOX is the greatest thing since Nutella, but how does it work?"

The key to implementing a multilayer data architecture, and hence OOX, is to be able to process an XML data document serially through two or more XSL transformations. In our corporate document example, to present the list of press release headlines we need to perform two transformations. First, we must transform the press releases document into a generic corporate document, which will constitute the middle layer of data. Then, we need to perform a second transformation to render the header content from the generic document as XHTML in a browser. This sounds simple enough, but the central issue becomes figuring out how to perform both transformations in a single step.

As I became more interested in pursuing an OOX approach for my own Web development, I was surprised to discover that I couldn't find a direct method for performing multiple transformations on an XML data object. Since I had no desire to become an XSL contortionist for each new Web project, I set out to develop a usable tool for performing multiple serial transformations in a single step. Ideally, I wanted a pure XSL solution that would blend seamlessly into an XML content management framework.

The end result of this effort was the creation of an original XSL transformation that makes it possible to invoke multiple serial transformations by passing a single XML document to the browser. In essence, this new transformation is a controller, which operates against an XML script. The script specifies the source content document and the various transformations it should undergo. Thus, the controller takes the source document specified by the script and iteratively applies each trans-

formation listed in the script (see Figure 3). The controller XSLT and all of the files required for reproducing the examples in this article are available at www.sys-con.com/xml/sourcec.cfm.

Using the Controller

As you may recall from our example, in order to render press release headlines in a browser, we need to process the press releases source document through two transformations. This is easy with the assistance of the controller. All we have to do is create an XML script that specifies the source data and transformations. This is accomplished using the tags <source> and <filter> respectively.

```
<?xml version="1.0" encoding="UTF-16"?>
<?xml-stylesheet type="text/xsl" href="controller.xsl"?>
<transformation>
  <source>white_papers</source>
  <filter>papers_to_generic</filter>
```

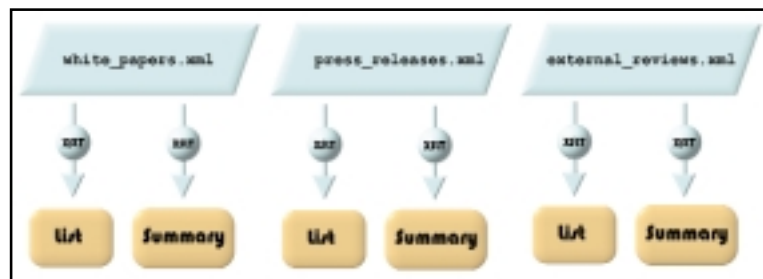


Figure 1 • Example of two-layer architecture

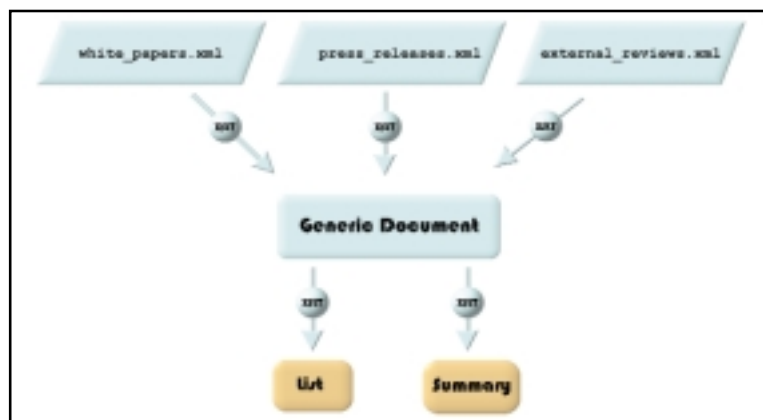


Figure 2 • Example of OOX architecture

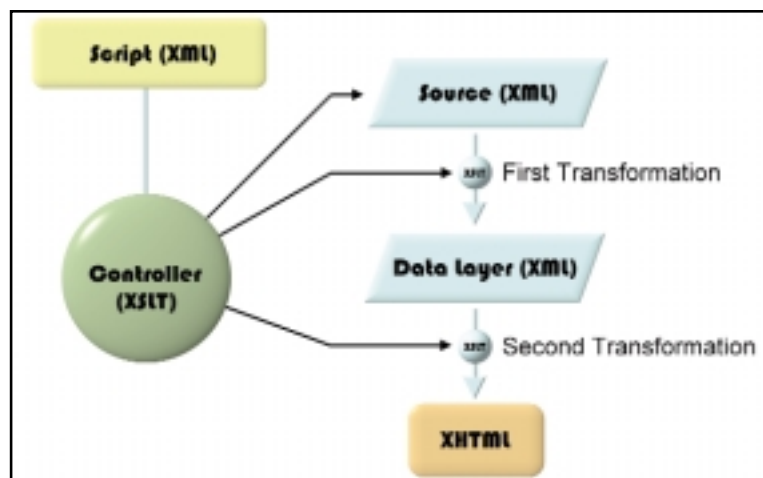


Figure 3 • Controller XSLT governs the flow of content

```
<filter>generic_to_summary</filter>
</transformation>
```

I decided to use the “filter” tag to refer to each individual transformation, and the <transformation> tag to refer comprehensively to the series of transformations that the source file undergoes. Thus, the <transformation> tag is used only once, much like the <body> tag in an HTML document. It is also worthwhile to note that the controller is actually the script’s stylesheet. If the script did not include a reference to the controller, the browser would not know how to process the script.

Step-by-step: the first transformation

When the browser’s XML parser applies the controller to the script, it performs each transformation consecutively in memory, using the output from one as the input to the next. Ultimately, the output from the final transformation is rendered in the browser. In our example, the first transformation converts the source document of press releases into the generic format. However, because this is not the final transformation, we are not privy to its output. So just to verify what is happening behind the scenes, I have invoked our first transformation directly from the source document. The output from this transformation is shown in Listing 1. Thanks to the controller, we normally would not see this generic data document, but it does occur in memory.

Step-by-step: the second transformation

Once the first transformation has taken place, the controller uses the resultant document (i.e., Listing 1) as an input to the second transformation. Since we are using a total of only two consecutive transformations, the browser renders the output from the second transformation, which is a formatted list of press release headlines (see Figure 4).

We can write a comparable script for external reviews, which will produce a similar result.

```
<?xml version="1.0" encoding="UTF-16"?>
<?xml-stylesheet type="text/xsl" href="controller.xml"?>
<transformation>
  <source>external_reviews</source>
  <filter>reviews_to_generic</filter>
  <filter>generic_to_list</filter>
</transformation>
```

Note that the second transformation, generic_to_list, is the

same for both of our scripts. This is because after we have transformed the original documents into a generic document, we can use the same transformation in both cases to render a formatted list of header content.

The important thing to remember in implementing OOX is that the scripts, which are XML data documents themselves, are used to initiate a process that ultimately produces an output in the browser window. In other words, if you want to present content in the browser, you need to reference the script that leads to the desired output. This can be confusing at first because most of us are accustomed to referencing content documents directly. Typically, if we want to display a list of items stored in an XML data document, we reference that document’s URI directly. The data is then presented according to the document’s stylesheet. In OOX, however, we take the indirect approach of referencing a script that details how the source content will be massaged before it is finally rendered.

‘Hello World’

Now that we’ve walked through an example at the conceptual level, you are probably eager to get your hands dirty and do some coding. As with HTML, you don’t need any special development environment to get started. A simple text editor like Notepad is sufficient, although a more sophisticated tool, such as Macromedia’s HomeSite, makes it easy to go back and forth between editing and testing. For a simple proof of concept, you will need a content file (XML), two transformations (XSL), the controller (XSL), and a script (XML). Once you’ve created these documents, you can render your content by entering the path of the script file in your browser. I encourage you to look at the example code, which has been tested in IE 6.0. Sometimes the quickest way to get started using a new technology is to take a working example and gradually modify it to suit your own needs.

Tip: In developing data-to-data transformations, it is a good idea to check the outputs at intermediate stages (as shown in Listing 1) to make sure that the abstract layers exhibit the intended data structure. Microsoft makes available a tool that allows you to see the result of an XSL transformation as XML formatted data. Once you install this tool, you can right-click on the document in Internet Explorer and select “View XSL Output”. However, this will only work if the XML file you are viewing in the browser specifies an XSL stylesheet. That means that even though the controller doesn’t require it, you will need to include a line in your source document such as

```
<?xml-stylesheet type="text/xsl"
href="my_transformation.xml"?>
```

for testing purposes. The package containing this tool is called “Internet Explorer Tools for Validating XML and Viewing XSLT Output” and is available online at <http://msdn.microsoft.com/library/default.asp?url=/downloads/list/xmlgeneral.asp>.

Using OOX to Manage Content Upgrading existing code

When it comes to Web development, I tend to learn things on a need-to-know basis. There’s simply too much innovation these days to keep track of all the emerging technologies. So if you’re like me, before you invest a lot of time in using OOX you will want a clear sense of how easily you can incorporate it into your work and how useful it will be.

The easiest way to start applying OOX is to retrofit an existing Web site. Here the goal is not to alter the functionality of the site, but to make its content more manageable. The key is

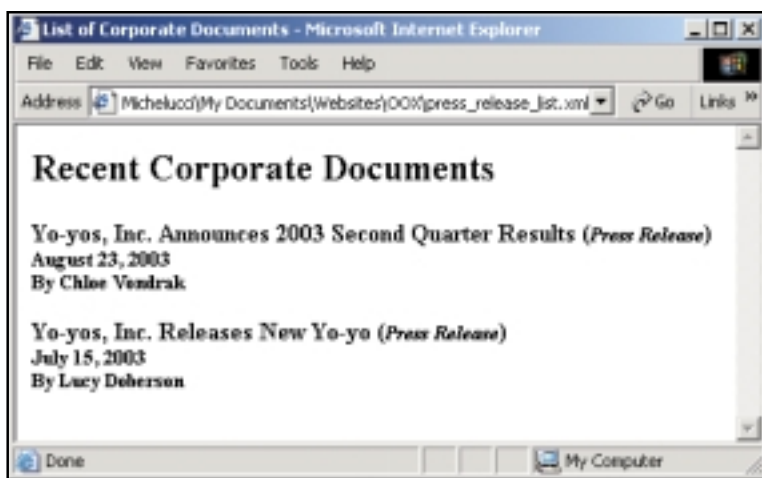


Figure 4 • The end result of two consecutive transformations

either to identify content elements that are presented in similar ways (as in the corporate documents example) or conversely, to recognize presentation elements that draw upon related content. To illustrate the latter case, consider the following. Many Web sites present a main menu horizontally at the top of the page, along with a submenu in the sidebar, which depends upon the main menu selection. Furthermore, these Web sites sometimes make available a site map to visitors as an alternate method of locating content. The site map typically provides a hierarchical list of hyperlinks for accessing pages directly. Since the menu elements and site map present related content, this Web layout can benefit from multilayer data abstraction.

Unlike the corporate document example in which we transformed specific documents into a generic format, in this example we begin with a generic data document (see Listing 2), which contains hierarchical information, and use transformations to achieve greater specificity. Since the site map is based upon the complete hierarchy of Web pages, we can directly transform the generic document into a site map.

The menu system, on the other hand, corresponds to only a subset of the Web site's page hierarchy. Specifically, the main menu corresponds to the first level of Web pages, and the submenu corresponds to the second hierarchical level. Therefore, rather than extracting those elements directly from the generic document and presenting the menus using single transformations, we can separate the process into two stages. The first transformation is used to create a new data document containing only the hierarchical information relevant to the menu system, that is, the first two levels.

```
<?xml version="1.0" encoding="UTF-16"?>
<menu>
  <main name="Products" link="products.xml">
    <sub name="Yo-yos" link="yo-yos.xml"/>
    <sub name="Marbles" link="marbles.xml"/>
  </main>
  ...
</menu>
```

This menu data document can then undergo one or more second-tier transformations to present the appropriate menu and submenu as needed.

The OOX architecture for this site map and menu implementation is depicted in Figure 5. In examining this architecture, you may notice the modularity inherent to this approach. The use of a middle layer for representing the menu data permits the reuse of the menu and submenu transformations in the context of other Web projects. To reuse those transformations for a Web site that doesn't have a site map you could revert to a two-layer design and create the menu data document manually.

In addition to gaining reusable transformation objects, another advantage of this implementation is that the hierarchy document becomes a one-stop shop for modifying the site map and menu systems. Adding a new submenu item is accomplished with a single entry in the hierarchy document. Once this new item has been inserted, the site map and the menu system will automatically reflect the new content.

As you have seen, retrofitting an existing Web site with an OOX architecture is based upon identifying related content groups (e.g., corporate documents) or presentation groups (e.g., the menus and site map). Each group defines a potential opportunity for improving content management. Therefore,

with a little planning, the task of overhauling a Web site with OOX can be an incremental and, hence, manageable process.

This gradual process of implementing architectural upgrades, such as those involving OOX, is called refactoring. Specifically, refactoring refers to making internal design changes that facilitate understanding and working with code without materially affecting its functionality. Martin Fowler introduced this term in his 1999 book, *Refactoring: Improving the Design of Existing Code*. It is sometimes difficult to justify refactoring when there is a competing need for new content or functionality. However, the piecemeal nature of refactoring with OOX makes it possible to enhance a Web site's architecture without disrupting ongoing development. In fact, there are compelling reasons for allocating resources to refactoring. For example, if you do want to add new functionality to a Web site, refactoring first can save time later by making the site more extensible. Furthermore, a timesaving benefit specific to Web sites (as opposed to software) is streamlined content management.

Starting a new project

If you are embarking upon a new Web project, the OOX framework can be instrumental in the design phase. This is because OOX unifies the goals of developing an architecture and preparing for content management. By taking this approach, outlined in Figure 6, you will ensure that your Web site is modular, extensible, and easily maintained. Begin by identifying the content and presentation domains for your Web

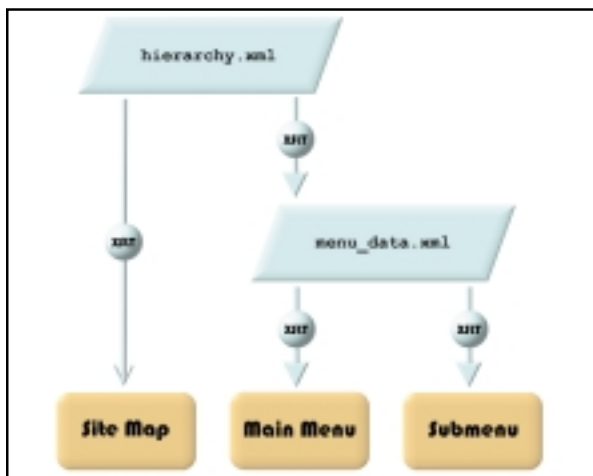


Figure 5 • OOX architecture for site map and menu implementation

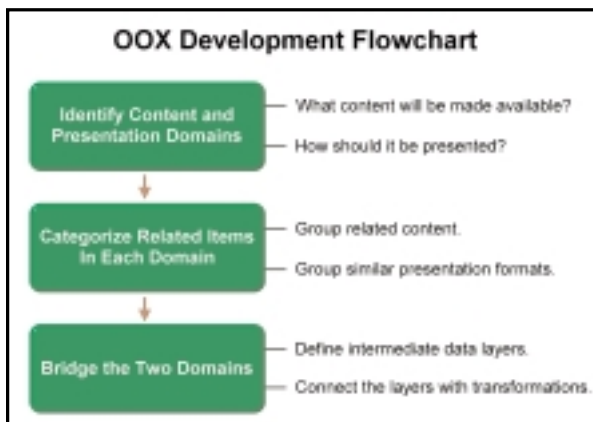


Figure 6 • OOX-based Web development

site. In other words, make a list of the content you will be providing, and then make another list of the various ways in which you will present that content to your site visitors. Ideally, the presentation list would be based upon a usability analysis. The next step, which may sound familiar by now, is to identify relationships within each domain so you can categorize items into groups. These groups form the basis for defining intermediate data layers. Finally, connect the content domain to the presentation domain using the intermediate data layers and appropriate XSL transformations.

There will be many plausible architectural configurations to choose from in most OOX-based endeavors. The key to a good design is parsimony. In other words, less is more. Be selective about adding data layers by making sure that all of your design decisions are based upon achieving a specific benefit, such as content management, reuse, or extensibility.

Once you've built your Web site using this approach, managing content is simply a matter of editing the data documents in the top layer. The controller transformation and script files will ensure that any content changes made to the top layer will trickle down to the presentation layer.


OOX: More Than Just a Bag of Tricks

OOX, like OOP, isn't just about stringing together multiple transformations, using extra data layers, or treating schemas like interfaces. It's an approach to content management and Web architecture that involves the judicious application of data abstraction and the reuse of transformation objects. When applied strategically, OOX can result in a low-maintenance Web site that is quickly built, logically organized, and robust to structural content changes.

As you are likely aware, there are feature-rich software

tools on the market to facilitate Web development and content management. Many of these tools function by storing proprietary metadata, which describe both structural and thematic aspects of the Web site. For example, metadata might be used to programmatically maintain navigation links on all pages of a Web site. These metadata are not directly accessible to the Web developer, so even though the software uses them internally for content management, they may impede fine-level control. Furthermore, migrating from one of these content management tools to another can prove vexing because the tools often do not recognize each other's metadata.

In contrast to most content management tools, OOX relies exclusively upon W3C-based technologies. Therefore, in adopting OOX as a Web development paradigm, it is possible to exercise complete control over your Web site without getting locked into proprietary technology. Furthermore, flexible tools can work in concert with OOX development.

OOX may not be suitable for all developers. But if you have dabbled in XML and aren't afraid to explore the power afforded by XSLT, you might be surprised at what the latest addition to alphabet soup has to offer for content management. 

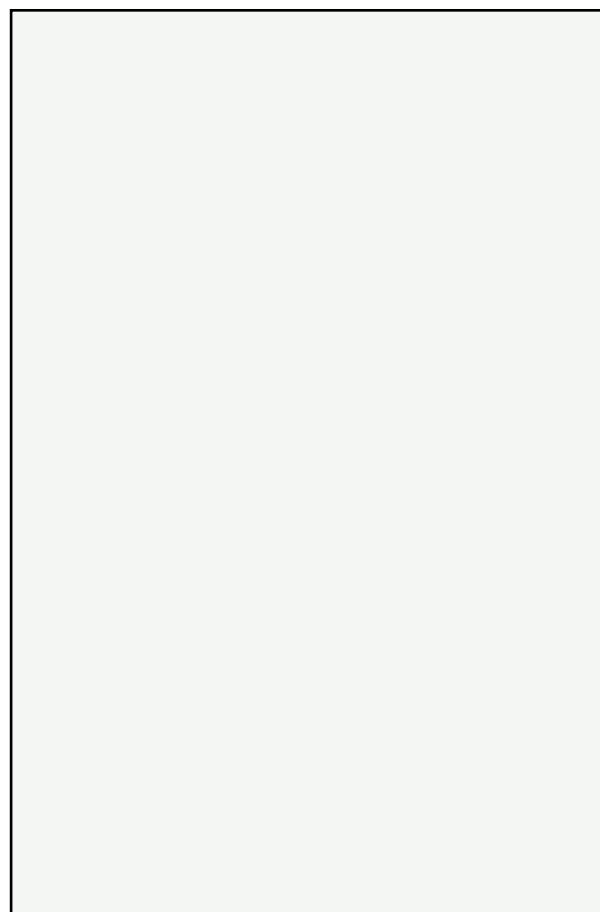
AUTHOR BIO

Pietro Michelucci directs the Experimental Data Institute, a software consultancy specializing in Web applications, data interoperability, and user interface design. In addition to delivering solutions to federal, Fortune 100, and small business clients, he builds R&D collaborations to address various public and private sector needs. His latest digression is the development of a Delphi API for an open-source, native XML database. He holds a joint Ph.D. from Indiana University.
www.experimentaldata.com

PEM@EXPERIMENTALDATA.COM

XML-J ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
Altova	http://xmlj.altova.com/xslt	978-816-1600	44
CTIA Wireless I.T.	www.ctiashow.com		25
Ektron	www.ektron.com/xmlj	603-594-0249	43
IBM	ibm.com/developerWorks/toolbox/seeit		2
Isavix Corporation	www.isavix.net	866-472-8849	9
LinuxWorld Magazine	www.sys-con.com	888-303-5282	37
Mindreef	www.mindreef.com	603-465-2204	4
SpeechTEK 2003	www.speechtek.com	859-278-2223	17
SYS-CON Media	www.sys-con.com	888-303-5282	27
Web Services Edge West 2003	www.sys-con.com	201-802-3069	31-36

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.



ctia Wireless I.T.

www.ctiashow.com



Content Management, XML, and the Promise of Web Services

Optimizing XML can make life simpler

Thriving organizations constantly explore new ways to share information via Internet sites, intranets, extranets, portals, CRM systems, and elsewhere. Organizations are finding it increasingly necessary to more carefully manage the process of creating, publishing, and reusing information.

How can companies:

- Improve searchability on Web sites, including large intranet sites, content-heavy Internet sites, etc.?
- Enhance opportunities to share content between employees, customers, partners, locations, etc.?
- Control and streamline the creation and presentation of news feeds, event listings, frequently asked questions, job listings, discussion forums, project updates, corporate messaging, and the like?

In the search for solutions, content management has emerged as a fast-growing sector of the IT industry.

Whether organizations have built or bought a solution, most have not yet gained the full value of a content management system (CMS). Many technical experts and IT business strategy decision-makers blindly believe that high cost, radical complexity, and long-winded implementations are prerequisites for content management projects. This is not true. Practical options exist for quick, flexible, scalable integration, and they do not necessarily cost an arm and a leg – even for projects with sophisticated needs.

Today, organizations are finding new ways to leverage XML within CMSs. The ability to reuse content is becoming a must-have service, particularly as handheld devices and print versions of Web content are needed anytime, anywhere.

All organizations, even those not currently using XML, should ensure that their CMS strategy includes forward-thinking plans for realizing the power and value of XML.

How Can XML Be Optimized in a CMS?

Business end-user XML authoring

Consider the evolution in word processing. First, word processing software allowed business users to apply their own formatting to paper documents. Previously, professional typesetters applied formatting “tags” to the document. Next, WYSIWYG HTML editors



allowed business users to add HTML formatting for Web content in an environment that feels just like word processing. With this functionality, IT departments overcome the challenges of Webmaster bottleneck. What's next? Business-user focused, WYSIWYG XML authoring in the browser is the next logical step.

A CMS should hide the complexity of XML from business users. The system

should offer a familiar, easy-to-understand environment that automatically applies XML tags to content “behind the scenes.” Cutting and pasting from Microsoft Word, for example, should be seamless. A Web developer or system administrator should be able to create radically intuitive “smart Web forms” that “transparently” structure and power the XML tagging process. The creation of such a form should be as simple as developing a template in Microsoft Word.

Strictly enforced content and page layout

A CMS becomes even more powerful if it takes full advantage of cascading style sheets (CSS). The deploying Web developer or system administrator can set parameters for styles and classes, thus forcing end users to comply with specific standards.

Use of CSS within an XML-enabled smart Web form extends this functionality even further. Within a CMS, XML can be used to validate content and to transform content for various consumers. By combining CSS, XSLTs, schemas, and DTDs, the content's style can be enforced for presentation on one device (such as the Web), then simply and automatically modified for alternative presentation on a different device (such as print, PDA, mobile phone, other servers, etc.). The CMS should allow for validation against a schema or DTD both locally and on the Internet. The system should also allow for handling and management of XSLTs within the application.

New possibilities with Web services and RSS

As Web services become more prevalent and standards evolve for content sharing, communication barriers

AUTHOR BIO

Bill Rogers is founder and CEO of Ektron, Inc. Ektron is a leader in developing flexible and affordable content management systems and Web authoring tools, including feature-rich HTML and XML content management systems that scale for any need, the industry's first “word processor on the Web,” and one of the first XML editors designed for business users.

between dissimilar platforms will disappear. Soon, Windows-based CM solutions will share content with non-Windows CM solutions and vice versa. With Web services, content management will take on a new face by truly managing information in a global environment.

A CMS that takes advantage of Web services can become an engine in a multitiered application. Organizations can more easily get information (text, images, data, etc.) into and out of the

Consider RSS, a known schema for transmitting and presenting content. A system is "fed" with an XML block and knows how to present it. Over time, RSS will be exposed as a Web service.

Now, consider content. Today's problem with content is that it has no well-defined schema. RSS is perhaps a start at moving in this direction. Today, organizations are hanging RSS feeds out on their Web sites as a URL. Take this one step further and the implications

zation initiatives are under way (Web Services Interactive Applications Group, Web Service for Remote Portals, etc.).

To me, content management is an engine and XML is one fuel for that engine. Until recently, the technology our companies created has focused primarily on getting content to the Web in the easiest, most efficient way possible. The lessons we've learned yield great value for creating an environment that utilizes content throughout the IT infrastructure in the easiest, most efficient way possible.

Many emerging technologies have interesting implications for content management – the tablet PC, speech recognition, wireless devices, and handwriting recognition, just to name a few. These will radically change creation of and access to content. Regardless of the way content is created or accessed, XML and Web services hold great promise for enhancing opportunities to retrieve, store, edit, publish, manage, and reuse content. XML will continue to simplify life for IT and non-IT professionals alike, especially in the area of content management. ☉

INFO@EKTRON.COM

"...content management is an engine and XML is one fuel for that engine"

CMS regardless of the platform on which it was built. As standards become more available, intelligence can be created within content management servers so they can locate and communicate with one another. The future will see a "content management box" sitting in an organization's IT infrastructure serving content to a variety of places including handhelds, Windows applications, other servers, etc.

could be significant. If the RSS standard (let's say version 2.0) is exposed as a Web service and people register their service with a UDDI server, then programs can query the UDDI registry and incorporate the feed programmatically. The potential is limitless.

We see certain industries beginning to use content schemas, for example mortgage companies. Many groups have been formed and many standardi-



THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

Go to www.SYS-CON.com

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

JAVA > Newsletter

WebServices > Newsletter

XML JOURNAL > Newsletter

wireless > Newsletter

XML Logic > Newsletter

WebSphere > Newsletter

COLOFUSION > Newsletter

.NET Journal > Newsletter

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

FREE

E-Newsletters

SIGN UP TODAY!

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




WRITTEN BY GREG WATSON

Using XSLT to Generate SQL

A simple, adaptable example to get you started

Examples of selecting data from a database to produce XML are not too difficult to find. A Google search or a visit to the XML section at your local bookstore will no doubt reveal several such examples. It can, however, be more difficult to locate examples of how to insert data from XML into a database. Even if you locate such examples, it can be a challenge (to say the least) to adapt them to large XML files in which the data you need resides in a selected number of elements nested several levels deep within the file.

This article illustrates how to use XSLT to generate SQL, which can then be loaded into an Oracle database using Unix shell scripts. For the purpose of illustrating the concept, the example given is relatively short and straightforward, but the method used in this article could easily be adapted to more complex projects. Furthermore, although this article focuses on getting data into an Oracle database from XML in a Unix environment, the concepts could be adapted to work with other database software running on other platforms.

There are at least two known methods for taking data from XML files and inserting it into Oracle. One method is to use the Java XSU insert method with an XML file. With this method, the structure of the XML file mirrors the structure of the Oracle table. Another method is to use Oracle's XML data type, which is available in Oracle9i. Using this method, a developer can insert the XML file directly into Oracle "as is." Both methods have their advantages in working with smaller XML files in which the elements are not heavily

nested. However, in working with files in which the elements are heavily nested, both methods require the generation of a secondary XML file for database loading. This is a requirement if you wish to select only certain elements from a large, heavily nested XML file. Using the method proposed in this article, generating a secondary XML file for database loading is not necessary, provided the needed data resides in the original XML file.

If you've worked with Oracle SQL*Plus, and with files having a .sql extension, the battle is half won. These files contain SQL code and are used to load data into Oracle in the SQL*Plus environment. In the same way that you use an XSLT processor – such as SAXON or Xalan – to generate XML or HTML output from XML, it is possible to use XSLT to generate a text file output. SAXON can be used to generate a text file having a .sql extension in the following manner: `java -jar saxon.jar -o file.sql file.xml file.xml`. This command may be appropriate to execute in this way if only a small number of files are to be processed. If there is a need to batch process a large number of files, it may be helpful to write a shell script to shorten the command syntax. To do this in a Unix environment, a shell script like the one below may be used. This script, which is called `gensql` in this article, is also referenced in the sample code for this article, available at www.sys-con.com/xml/sourcex.cfm.

```
#!/bin/sh

SQLFILE=$3
XMLFILE=$1
XSLFILE=$2
```

```
# Here we would test for the correct
number of arguments.

if [ $# != 3 ]
then
    echo "Usage: gensql file.xml
file.xml file.sql"
else
    /usr/bin/java -jar saxon.jar -o
$SQLFILE $XMLFILE $XSLFILE
fi
```

Before using this script to build a SQL file, you must first identify the elements within the XML file from which the data will be extracted. Even if you're relatively familiar with the data within the XML file, if it is a large file (or a group of large files), it may be helpful to build a table of values. Before addressing this table, it's helpful to describe the kind of data that will be used in our fictitious example.

In this example, consider the needs of a national coordinator working with high school baseball teams. To assist in this work, XML files must be built for each state in the U.S. containing the names of high schools, their location, where each team plays its home baseball games, and each team's mascot. For this example, consider the XML file for the state of Indiana focusing on three Indiana towns whose teams play each other: Littleville, Flatville, and Lillyville. For the database example, it's necessary to extract from the sample XML file the name of each town, the field where each team plays its home games, and each team's mascot. Table 1 is helpful in extracting this information.

Now the XSLT file may be built. In getting the SQL file output from the XML file, "text" must be specified as the out-

AUTHOR BIO

Greg Watson is a computer systems analyst working in the area of XML development at the Defense Intelligence Agency's Missile and Space Intelligence Center. In 2002, he spoke at the Intelligence Community Conference on XML Metadata.

put type within the XSLT file in the following manner:

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  version="1.0">
<xsl:output method="text"/>
```

Next, the root level element of the XML document must be matched with the following:

```
<xsl:template match="/">
```

Although the XML example file focuses on the state of Indiana, XML files also exist for other states in the U.S., and the values for each state must be differentiated. At this point, the top-level element must match an attribute value of "IN", which is the two-letter U.S. abbreviation for Indiana. Although the sample XSLT file will contain only values associated with Indiana, information could be retrieved about other states by repeating this same if statement for each state. After entering the INSERT statements for each state, the if statement for the values associated with the state in question is closed. There may be several INSERT statements for a particular state, but by matching the two-letter abbreviation for that state in the XSLT file, all the values for a state will be selected from an XML file containing only data about that state. Thus, there may exist one XML file for each state, but only one XSLT file for all 50 states.

```
<xsl:if test="/State/@id='IN'">
```

At this point in building the XSLT file, you're ready to add the code necessary for building the SQL INSERT statement. The values to insert into the Oracle table will be the ones identified earlier in the sample table. When extracting these values for the sample data, the XSLT normalize-space function is used. This function will remove leading and trailing spaces from the data contained within the XML tags. This function is used to limit the number of characters allowable in each column in the database table. The XML tag names in our sample XML file correspond directly to the column names in our sample Oracle table, so we would build the XSLT code for our SQL INSERT statement in the following manner:

```
INSERT INTO INDIANA
(TOWN, FIELD, MASCOT)
```

```
VALUES
('<xsl:value-of select="normalize-
space(//HighSchool[@Number='01']/Town
)'>',
'<xsl:value-of select="normalize-
space(//HighSchool[@Number='01']/Field
)'>',
'<xsl:value-of select="normalize-
space(//HighSchool[@Number='01']/Mas-
cot)'>');
```

In writing the XSLT file for the sample data, you'd need to repeat this same INSERT statement three times in order to get all the data from the XML file into the SQL file. To select the additional elements from the XML file, you'd simply replace the [Number='01'] above in the XSLT example with [Number='02'] and then [Number='03'].

Instead of using a DTD (as done in the sample code), an XML Schema could be used. In using an XML Schema a developer could possibly avoid having to use the normalize-space function present in the XSLT file. The data in the XML file could be constrained using an XML Schema, which would eliminate leading and trailing spaces. While schemas have their advantages and are effective for constraining data in some XML files, they are ineffective in constraining elements that need to allow for mixed content. In other words, in using an XML Schema, a developer cannot constrain the value of an element that would allow for the ability to insert other elements (such as footnote reference elements) within that element tag. Generating a separate XML file using XSLT can solve this problem. With this separate file, you can then constrain the content in that XML file using a schema. This, however, creates a separate file that must be managed in some way. Thus, in cases where the data is known and in cases in which mixed content must be allowed in the XML files, it is best to use a DTD and constrain the data in an XSLT file.

The last step in the process is to load the SQL file into Oracle. The simplest way to do this would be to log manually into SQL*Plus and execute the @file.sql

command at the SQL prompt. This solution may be fine in cases in which you have a small amount of data to load. But in cases in which you need to batch load a larger amount of data, it is best to load the data through a script. Sample Unix scripts are included for loading the SQL file into Oracle in the sample code.

In the Unix scripts, the following main steps load the XML data into Oracle:

1. Delete the old SQL files generated from previous data loads.

“Examples of selecting data from a database to produce XML are not too difficult to find”

2. Create a new SQL file from our XML with the gensql script (mentioned earlier).
3. Create an empty file with a .sql extension.
4. Append the TRUNCATE TABLE command to the file created in Step 3. (This command will empty the table but leave the table structure intact.)
5. Append the file created in Step 2 to the file created in Step 3.
6. Append the “exit” command to the file created in Step 3.
7. Execute the remote hostname command (rsh) to the Oracle Unix server. When executing this command, execute the Unix script containing the following command: sqlplus userid/password @SQLFileCreatedIn-StepThree. Within the script containing the sqlplus command, it may be necessary to set ORACLE environment variables to their value on the remote Oracle Unix server.

These steps may seem somewhat involved, but they represent a direct way to load XML data into Oracle. The main factor to watch out for in imple-

XML tag name	Values from the XML file to be inserted into the Oracle table		
Town	Littleville	Flatville	Lillyville
Field	Tiny Diamond	Flat Earth	Fresh Air
Mascot	The Ants	The Pancakes	The Lilies

Table 1 • XML file information

menting these steps is making sure the ORACLE environment variables are set properly. Depending on how your Unix environment is set up, it may be unnecessary to set these environment variables in the script. If you have an Oracle DBA or access to an Oracle power user on site, it would be best to get their advice on setting the environment variables for the script. If you have to fend for yourself in set-

“There are at least two known methods for taking data from XML files and inserting it into Oracle”

ting the variables, log in to the Oracle Unix server, run the “env” command, and note the value of the ORACLE_ environment variables. Then log out of the remote Oracle server and run the env command locally. After doing this, compare the value of the ORACLE environment variables on the remote server with those on the local server. If the variables are set differently, try setting them in the sqlplus login script to their value on the remote Oracle server.

Summary

I’ve looked mainly at how to generate a SQL file from XML using XSLT, and then how to take this file and load it into an Oracle database using Unix scripts. I’ve also compared this method of loading data from XML files into Oracle with other methods well-known to anyone required to work between XML files and a database, and I’ve considered the advantages and disadvantages of using XML Schemas to constrain data within XML element tags.

In closing, I hope that the example in this article is helpful. Any questions about the example or any of the sample code may be directed to me via e-mail; I’ll be happy to help. ☎

References

- “How to Specify Output File Extension in Saxon.” www.biglist.com/lists/xsl-list/archives/200304/msg01412.html
- SAXON XSLT Processor: <http://saxon.sourceforge.net>
- Kay, M. (2001). *XSLT Programmer’s Reference*, 2nd edition. Wrox.
- Naude, F “Oracle SQL*Plus FAQ.” www.orafaq.net/faq-plus.htm
- Naude, F “Oracle XML FAQ.” www.orafaq.net/faqxml.htm
- Sobell, M. (1995). *A Practical Guide to the Unix System*, 3rd edition. Addison-Wesley.
- Sobell, M. (1999). *A Practical Guide to SOLARIS*. Addison-Wesley.

DIWATGB@MSIC.DIA.MIL



WEB SERVICES

Building Manageable Web Services from the Ground Up

The management criteria developers must consider



PROJECT MANAGEMENT

Managing and Documenting Your Project XML Style

Some potentially powerful uses for XML technology



REUSE

Reusable Visual Controls Using XML

A consistent look-and-feel throughout your application



JAXB AND JDO

A Perfect Database Round-Trip Using JAXB and JDO

Take advantage of the complex work done underneath the application layer

DON'T
MISS
XML-J
SEPTEMBER

REAL-WORLD SOLUTIONS

International Web Services Conference & Expo

Web Services Edge

3rd Annual

web services **EDGE**
conference & expo

Delivering
.NET, JAVA,
Mac OS X,
& XML
Technologies



2003 WEST



SEPT. 30 - OCT. 2, 2003
Santa Clara, CA

web services **EDGE**
conference & expo

**REGISTER
TODAY!**

CALL 201-802-3058
www.sys-con.com

Register before August 1st and

**SAVE
Up To \$300**

Produced and Presented:

**SYS-CON
EVENTS**

Media Sponsors:

WebServices
JOURNAL

JAVA DEVELOPERS
JOURNAL

WebSphere
DEVELOPERS JOURNAL

XML JOURNAL

.NET JOURNAL

WebLogic

LinuxWorld
JOURNAL

wireless
JOURNAL

LINUXWEEK

CE Advisor

ColdFusion
DEVELOPERS JOURNAL

PowerBuilder DEVELOPERS
JOURNAL

market WIRE

HSP STREET
JOURNAL

OASIS

SDTimes

LINUXWORLD.COM

SAMS

Real Industry

WEB SERVICES EDGE CONFERENCE & EXPO

web
services
conference & expo

EDGE
& expo

SEPT. 30 - OCT. 2, 2003
Santa Clara Convention Center

It is our pleasure to bring the latest edition of the highly successful Web Services Edge Conference & Expo to the Santa Clara Convention Center, September 30 - October 2, 2003. The Third Annual Web Services Edge West Conference & Expo will continue to build on our past success to make available the most current and relevant information to you, our valued attendee.

With the widespread adoption of Web services across the industry, developers are facing new challenges. In this year's conference program, we will address these challenges with our most comprehensive program to date. Web Services Edge 2003 West will provide practical approaches and solutions to overcome the hurdles in developing and deploying Web services in today's competitive markets. Once again Web Services Edge will feature dedicated tracks covering Java, Web Services, .NET, and XML - along with the newly added Mac OS X Track. Sessions in this track will highlight the use of the Mac OS X platform, which combines the ease of use of a Mac with the power of Unix, in applications and Web services development, deployment, and management.

Your three days will include highly informative keynotes, conference sessions, tutorials, industry-leading university certification programs, case studies, and demo presentations. The Expo Hall will be open on September 30 and October 1, featuring the largest grouping of quality exhibitors prepared to field your questions and solve your development needs.

All the best,
SYS-CON Events

FEATURES & ATTRACTIONS TO ALL CONFERENCE & EXPO REGISTRANTS

- 3 Days Packed with Education and Training
- Keynotes & Panel Discussions from Industry Leaders
- 60 Hard-hitting and Informative Seminars
- FREE .NET Tutorial with Russ' Tool Shed
- Java University Certification Training
- Industry-Leading Certification Programs
- "Birds of a Feather" Discussions
- Round Table Discussions
- Opening Day Welcome Reception
- SAMS Meet the Authors Hot Topics Lounge
- Compelling Case Studies & Best Practices
- Hands-On Labs
- Featured Product Demonstrations
- Exhibit Floor featuring more than 40 companies and hundreds of products
- Real-time SYS-CON Radio Interviews

WHO SHOULD ATTEND

- Software Developer
- Software Engineer
- Development Manager
- Application Developer
- Technical Director
- Analyst/Programmer
- IT Manager
- Technical Architect
- Team Leader
- Software Consultant

HIGHLIGHTED SPEAKERS

Allan Vermeulen

CTO, Amazon.com

amazon.com



CTO and vice president at Amazon.com directly oversees the Platform Technologies group. This group is responsible for guiding Amazon.com's technology architecture, including building and acquiring foundational components. Prior to his move to Amazon.com, Vermeulen was CTO and vice president of development at Rogue Wave Software. He holds a PhD in Systems Design Engineering from the University of Waterloo.

John Schmidt

Leader of Systems Integration and Middleware, Best Buy Co.



John Schmidt is the chairman of the Methodology Committee for the EAI Industry Consortium and leader of systems integration and middleware at Best Buy Co., a leading specialty retailer of consumer electronics, personal computers, entertainment software, and appliances.

Dave Chappell

VP, Chief Technology Evangelist, Sonic Software



Dave Chappell is the vice president and chief technology evangelist for Sonic Software. He has more than 18 years of industry experience building software tools and infrastructure for application developers, spanning all aspects of R&D, sales, marketing, and support services. Dave has also been published in numerous technical journals, and is currently writing a series of contributed articles for *Java Developer's Journal*.

Anne Thomas Manes

Research Director, Burton Group



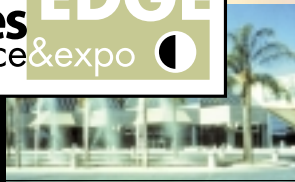
Anne Thomas Manes is a research director at Burton Group, a research, consulting, and advisory firm. Anne leads research for the Application Platform Strategies service. Named one of NetworkWorld's "50 Most Powerful People in Networking" in 2002, and one of Enterprise Systems Journal's "Power 100 IT Leaders" in 2001, Anne is a renowned technologist in the Web services space. Anne participates in standards development at W3C and OASIS. She is a frequent speaker at trade shows and author of numerous articles and the book *Web Services: A Manager's Guide*.

Register Online at

www.sys-con.com

WEB SERVICES EDGE CONFERENCE & EXPO

web
services
conference & expo



SEPT. 30 - OCT. 2, 2003
Santa Clara Convention Center

WEB SERVICES TECHNOLOGY

Presentations will include discussions of security, interoperability, the role of UDDI, progress of the standards-making bodies, SOAP, and BPM. Case studies cover the design and deployment of Web services in the marketplace.

Sessions will focus on:

- Interoperability
- Enterprise Networks
- Web Services Management
- Web Services Standards
- Web Services Orchestration
- Security (WS-Security, SAML)
- BPET4WS
- UDDI: Dead or Alive?
- ebXML & Web Services
- EAI & Web Services
- RPC vs Documents: Uses and Differences
- User Interfaces for Web Services
- Web Services Best Practices
- Service Oriented Architecture



XML TECHNOLOGY

Presentations will focus on the various facets of XML technologies as they are applied to solving business computing problems. Sessions will include emerging standards in XML Schemas, XML repositories, industry applications of XML, applying XML for building Web services applications, XML/XSLT/XQuery-based programming using Java/.NET, XML databases, XML tools and servers, XML-based messaging, and the issues related to applying XML in B2B/EAI applications. The XML Track is geared for audiences ranging from beginners to system architects and advanced developers.

Sessions will focus on:

- XML Standards & Vocabularies
- Introduction to XForms
- Securing Your XML and Web Services Infrastructure
- XQuery Fundamentals: Key Ingredient to Enterprise Information Integration
- XML and Enterprise Architecture: Technology Trends
- Standards-Based Enterprise Middleware Using XML/Web Services
- XML and Financial Services
- Canonical Documents for Your Business: Design Strategies
- XPath/XSLT 2.0: What's New?
- XML Schema Best Practices
- XML in EAI, Enterprise Portals, Content Management



MAC OS X

OS X represents a new wave of operating systems. It combines the ease of use of a Mac with the power of Unix. Sessions in this track will highlight the use of the Mac OS X platform in applications and Web services development, deployment and management.

Sessions will focus on:

- Introducing OS X (Panther): What's New?
- Quick Applications using AppleScript
- Enterprise Java and OS X
- Developing Web Services Using WebObjects
- Xserve: Ease of OS X and Power of Unix
- Introducing Quartz: 2D Graphics for Apple
- OS X for the Unix Developer
- Securing OS X Applications
- Java and OS X: A Perfect Marriage
- Programming Rich User Interfaces Using Cocoa



JAVA TECHNOLOGY

The Java Track features presentations aimed at the beginner, as well as the seasoned Java developer. Sessions will explore the whole spectrum of Java, focusing on J2EE, application architecture, EJB, and J2ME. In addition the track will cover the latest in SWT, Ant, JUnit, open source frameworks, as well as an in-depth look into the vital role that Java is playing in building and deploying Web services.

Sessions will focus on:

- Enterprise Java 1.4
- Ant Applied in "Real World" Web Services
- Developing Application Frameworks w/SWT
- Empowering Java and RSS for Blogging
- JUnit: Testing Your Java with JUnit
- JDK1.5: The Tiger
- Simplifying J2EE Applications
- Using IBM's Emerging Technologies Toolkit (ETTK)
- Apache Axis
- Meeting the Challenges of J2ME Development
- Integrating Java + .NET
- Squeezing Java



.NET TECHNOLOGY

Presentations will explore the Microsoft .NET platform for Web services. To the average developer, it represents an entirely new approach to creating software for the Microsoft platform. What's more, .NET development products - such as Visual Studio .NET - now bring the power of drag-and-drop, GUI-based programming to such diverse platforms as the Web and mobile devices.

Sessions will focus on:

- ASP.NET
- Security
- VB.NET
- .NET and XML
- Smart Device Extensions for VS.NET
- Best Practices
- Shared Source CLI
- .NET Remoting
- Smart Devices in Health Care Settings
- Mobile Internet Toolkit
- ROTOR
- Portable .NET
- ASP.NET Using Mono
- Using WSE with IBM's WSTK
- GUI applications Using Mono
- Portals - Windows SharePoint Services/SharePoint Portal Server
- Windows Server 2003 and IIS 6
- .NET and Java Interoperability
- Distributed .NET for Financial Applications
- Developing C# with Eclipse



For more information visit
www.sys-con.com
or call
201 802-3069

**SPECIAL
DISCOUNTS
AVAILABLE**

Take advantage of the Early Bird and Pre-registration values available right now, or save even more with a group of 5 or more. For special group discounts contact Michael Lynch at mike@sys-con.com, or by phone at (201) 802-3058.

Conference at-a-Glance

		JAVA	.NET	WEB SERVICES	XML
TUESDAY, SEPTEMBER 30 DAY 1	8:00AM – 4:00PM	REGISTRATION			
	9:00AM – 9:50AM	Enterprise Java 1.4	Using WSE 2.0	Web Services Management	Introduction to Xforms
	10:00AM – 10:50AM	Opening Keynote - Allen Vermeulen, CTO, Amazon.com			
	11:00AM – 6:00PM	EXPO OPEN			
	2:00PM – 2:50PM	Keynote Panel Discussion - Enterprise Application Integration			
	3:00PM – 3:50PM	Ant Applied in "Real World" Web Services	Smart Devices in Health Care Settings	Service Oriented Architecture	Securing Your XML and Web Services Infrastructure
	4:00PM – 4:50PM	Developing Application Frameworks with SWT	Using the Mobile Internet Toolkit	Web Services Orchestration	XQuery Fundamentals: Key Ingredient to Enterprise Information Integration
	5:00PM	OPENING NIGHT RECEPTION			
WEDNESDAY, OCTOBER 1 DAY 2	8:00AM – 4:00PM	REGISTRATION			
	9:00AM – 9:50AM	Integrating Java and .NET	Introduction to ROTOR	Security (WS-Security, SAML)	Standards-Based Enterprise Middleware Using XML/Web Services
	10:00AM – 10:50AM	Morning Keynote			
	11:00AM – 4:00PM	EXPO OPEN			
	2:00PM – 2:50PM	Keynote Panel Discussion - Interoperability: Is Web Services Delivering?			
	3:00PM – 3:50PM	JUnit: Testing Your Java with JUnit	Using Portable .NET	WS-BPEL	XML and Enterprise Architecture: Technology Trends
	4:00PM – 4:50PM	JDK1.5: The Tiger	ASP.NET with Mono	UDDI: Dead or Alive?	Using XML Schemas Effectively in WSDL Design
	5:00PM – 6:00PM	Squeezing Java	Using WSE with IBM's WSTK	Web Services Choreography, Management, and Security - Can They Dance Together?	Canonical Documents for Your Business: Design Strategies
THURSDAY, OCTOBER 2 DAY 3	8:00AM – 4:00PM	REGISTRATION			
	9:00AM – 9:50AM	Using IBM's Emerging Technologies Toolkit (ETTK)	Distributed .NET for Financial Applications	eAI & Web Services	XML and the Fortune 500
	10:00AM – 10:50AM	Morning Technical Keynote			
	11:00AM – 11:50AM	Apache Axis	Developing C# with Eclipse	RPC vs Documents: Uses and Differences	XPath/XSLT 2.0: What's New?
	12:00PM	BREAK			
	1:00PM – 1:50PM	Meeting the Challenges of J2ME Development	Windows SharePoint Services	The Seven Habits of Highly Effective Enterprise Service Buses (ESBs)	ebXML & Web Services
	2:00PM – 2:50PM	Keynote Panel Discussion - Summit on Web Services Standards			
	3:00PM – 3:50PM	Empowering Java and RSS for Blogging	BizTalk 2003	See www.sys-con.com for more information	See www.sys-con.com for more information
	4:00PM – 5:00PM	See www.sys-con.com for more information	See www.sys-con.com for more information	See www.sys-con.com for more information	See www.sys-con.com for more information

MAC OS X	
	Introducing OS X (Panther) What's New?
	Programming Rich User Interfaces Using Cocoa
	Quick Applications using AppleScript
	Java and OS X: A Perfect Marriage
	Enterprise Java and OS X
	Developing Web Services Using WebObjects
	Cocoa, Carbon, Java: Application Frameworks for OS X (When to use what)
	Securing OS X Applications
	Xserve: Ease of OS X and Power of Unix
	OS X for the Unix Developer
	Introducing Quartz: 2D Graphics for Apple
	See www.sys-con.com for more information

PROGRAM SUBJECT TO CHANGE



Russ' TOOLSHED

TINKERING WITH VISUAL STUDIO

FREE
Microsoft®
Tutorial

RUSS' TOOL SHED

Join Russ as he shows you how to use Visual Studio .NET

INTRO TO WEB SERVICES USING VS.NET

One of the key ideas behind the .NET strategy is the concept of software as a service, or in short, Web services. This session will explain what a Web service is and provide an overview of its related technologies like XML, SOAP, and UDDI. We will demonstrate how the .NET Framework makes it easy to implement them for new and existing applications. This session will also provide concrete best practices for building XML Web services using Visual Studio .NET. We'll answer many common questions like: How will my Web service scale? How can my XML Web services enable interoperability with Web services from other vendors as well as within my own organization? We'll delve into building highly reliable and secure Web services. Also, we will discuss issues such as dealing with complex data types using WSDL (Web Services Description Language), as well as securing SOAP messages using encryption. We'll see how developers can use enterprise-level XML Web services to simplify customer solutions.



ADVANCED WEB SERVICES USING ASP.NET

This session will explore some of the more advanced areas of SOAP in ASP.NET's support for Web services. ASP.NET Web services are the preferred way for Web developers to expose Web services on the Internet. The goal is quick, easy, and high-performing SOAP services. We will look at how to use the SOAP extension classes to create some very interesting applications on top of the core SOAP architecture found within the .NET Framework. For instance, you can implement an encryption algorithm or screen scraping on top of the Web service call. We'll dig into more advanced topics, explore the SOAP headers, and see ways to ensure security in our Web services.

.NET REMOTING ESSENTIALS

Microsoft .NET Remoting is the .NET technology that allows you to easily and quickly build distributed applications. All of the application components can be on one computer or they can be on multiple computers around the world. .NET Remoting allows client applications to use objects in other processes on the same computer or on any other computer to which it can connect over its network. During this presentation we will discuss what you will need to know to get started with .NET Remoting. We will talk about how .NET Remoting compares with DCOM, how to host remotable objects in a variety of applications, how to call remotable objects from a client application, how to control the lifetime of remotable objects, and how to secure remoting applications.

Register Online at
www.sys-con.com

CONFERENCE: Sept. 30 – Oct. 2, 2003 EXPO: Sept. 30 – Oct. 1, 2003**Santa Clara Convention Center • Santa Clara, CA****THREE WAYS TO REGISTER FOR CONFERENCE****1) On the Web:** Credit Cards or "Bill Me." Please make checks payable to SYS-CON Events.**2) By Fax:** Credit Cards or "Bill Me" 201-782-9651**3) By Mail:** 135 Chestnut Ridge Road, Montvale, New Jersey 07645, Attention: Registration**Please note: Registrations are not confirmed until payment is received.****Please complete sections 1, 2, 3 and 4****1 YOUR INFORMATION** (Please Print) ☐ Mr. ☐ Ms.

First Name _____ Last Name _____

Title _____

Company _____

Street _____

Mail Stop _____

City _____

State _____ Zip _____ Country _____

Phone _____

Fax _____ E-Mail _____

2 PAYMENT METHOD: (Payment in full due with registration)☐ Check or Money Order Enclosed (Registration confirmed upon receipt of payment)

Check # _____ Amount of Check \$ _____

Charge my ☐ Visa ☐ MasterCard ☐ American Express ☐ Discover

Name on card _____

Card # _____ Exp. Date _____

Signature _____

Billing Address (if different from mailing address) _____

3 PLEASE INDICATE**YOUR CONFERENCE CHOICE****Total Registration fee \$** _____☐ **GP Gold Passport** Good for all three days of the .NET, **Before 8/1/03 \$1,195.00 Before 9/26/03 \$1,395.00 Onsite \$1,495.00**Web Services, XML, Java, and Mac OS X Tracks, including
Keynotes, Panel Discussions, preferred seating for Microsoft
.NET's Russ "Tool Shed" Tutorial, and your choice of one Sun
Microsystems Java™ University Class☐ **3D Three Day Conference** **\$1,095.00 \$1,295.00 \$1,395.00**
(Does not include Sun Java™ Education)☐ **2D Two Day Conference** (Does not include Sun Java™ **\$995.00 \$1,195.00 \$1,295.00**
Education) (select any two days: ☐ Tue. ☐ Wed. ☐ Thurs.)☐ **1D One Day Conference** (Does not include Sun Java™ Education) **\$495.00 \$595.00 \$695.00**
(select any one day: ☐ Tue. ☐ Wed. ☐ Thurs.)☐ **JU1 Sun Java™ University Class** **\$595.00 \$695.00 \$795.00**
Select one: ☐ Web Services Programming
Using Java™ Technology and XML (Sept. 30)
☐ Java™ Fast Path: Programmer (Oct. 1)
☐ Java™ Fast Path: Architect (Oct. 2)☐ **JU2 Sun Java™ University Class** **\$1,095.00 \$1,295.00 \$1,395.00**
Select two: ☐ Web Services Programming
Using Java™ Technology and XML (Sept. 30)
☐ Java™ Fast Path: Programmer (Oct. 1)
☐ Java™ Fast Path: Architect (Oct. 2)☐ **JU3 Sun Java™ University Class** **\$1,195.00 \$1,395.00 \$1,495.00**
Select three: ☐ Web Services Programming
Using Java™ Technology and XML (Sept. 30)
☐ Java™ Fast Path: Programmer (Oct. 1)
☐ Java™ Fast Path: Architect (Oct. 2)☐ **EO Expo Only** **FREE FREE \$50.00****4****A. Your Job Title**

- ☐ CTO, CIO, VP, Chief Architect
☐ Software Development Director/Manager/Evangelist
☐ IT Director/Manager
☐ Project Manager/Project Leader/Group Leader
☐ Software Architect/Systems Analyst
☐ Application Programmer/Evangelist
☐ Database Administrator/Programmer
☐ Software Developer/Systems Integrator/Consultant
☐ Web Programmer
☐ CEO/COO/President/Chairman/Owner/Partner
☐ VP/Director/Manager Marketing, Sales
☐ VP/Director/Manager of Product Development
☐ General Division Manager/Department Manager
☐ Other (please specify) _____

B. Business/Industry

- ☐ Computer Software ☐ Government/Military/Aerospace
☐ Computer Hardware and Electronics ☐ Health Care/Medical
☐ Computer Networking & Telecommunications ☐ Insurance/Legal
☐ Internet/Web/E-commerce ☐ Education
☐ Consulting & Systems Integrator ☐ Utilities
☐ Financial Services ☐ Architecture/Construction/Real Estate
☐ Manufacturing ☐ Agriculture
☐ Wholesale/Retail/Distribution ☐ Nonprofit/Religious
☐ Transportation ☐ Other (please specify) _____
☐ Travel/Hospitality

C. Total Number of Employees at Your Location and Entire Organization (check all that apply):

	Location	Company
10,000 or more	01 <input type="checkbox"/>	01 <input type="checkbox"/>
5,000 - 9,999	02 <input type="checkbox"/>	02 <input type="checkbox"/>
1,000 - 4,999	03 <input type="checkbox"/>	03 <input type="checkbox"/>
500 - 999	04 <input type="checkbox"/>	04 <input type="checkbox"/>
100-499	05 <input type="checkbox"/>	05 <input type="checkbox"/>
100 or less	06 <input type="checkbox"/>	06 <input type="checkbox"/>

D. Please indicate the value of communications and computer products and services that you recommend, buy, specify, or approve over the course of one year:

- ☐ \$10 million or more ☐ \$10,000 - \$99,999
☐ \$1 million - \$9.9 million ☐ Less than \$10,000
☐ \$500,000 - \$999,999 ☐ Don't know
☐ \$100,000 - \$499,999

E. What is your company's gross annual revenue?

- ☐ \$10 billion or more ☐ \$1 million - \$9.9 million
☐ \$1 billion - \$9.9 billion ☐ Less than \$1 million
☐ \$100 million - \$999 million ☐ Don't know
☐ \$10 million - \$99.9 million

F. Do you recommend, specify, evaluate, approve or purchase wireless products or services for your organization?01 ☐ Yes 02 ☐ No**G. Which of the following products, services, and/or technologies do you currently approve, specify or recommend the purchase of?**

- ☐ Application Servers
☐ Web Servers
☐ Server Side Hardware
☐ Client Side Hardware
☐ Wireless Device Hardware
☐ Databases
☐ Java IDEs
☐ Class Libraries
☐ Software Testing Tools
☐ Web Testing Tools
☐ Modeling Tools
☐ Team Development Tools
☐ Installation Tools
☐ Frameworks
☐ Database Access Tools / JDBC Devices
☐ Application Integration Tools
☐ Enterprise Development Tool Suites
☐ Messaging Tools
☐ Reporting Tools
☐ Debugging Tools
☐ Virtual Machines
☐ Wireless Development Tools
☐ XML Tools
☐ Web Services Development Toolkits
☐ Professional Training Services
☐ Other [Please Specify] _____

SYS-CON Events, Inc. and SYS-CON Media make no warranties regarding content, speakers, or attendance. The opinions of speakers, exhibitors and sponsors do not reflect the opinion of SYS-CON Events and SYS-CON Media and no endorsement of speakers, exhibitors, companies, products, or sponsors is implied.



If you require special assistance covered under the Americans with Disabilities Act, please call 201-802-3058 by September 16, 2003.



CANCELLATIONS, SUBSTITUTIONS, REFUNDS
 Fax written request to SYS-CON Registration 201-782-9651. Requests for refunds received prior to August 29, 2003 will be honored, less a 10% handling charge; requests received after August 29, 2003, and before September 12,

2003, will be honored less a 20% handling charge. No requests for refunds will be honored after September 12, 2003. Requests for substitutions must be made in writing prior to September 26, 2003. No one under 18 is permitted to attend. No warranties are made regarding the content of sessions or materials.

Speakers, sessions, and schedule are subject to change without prior notice.

No solicitation by anyone other than official exhibitors, sponsors or marketing partners is permitted. Such behavior is cause for expulsion without refund.

Web Services Edge 2003

SEPT. 30 - OCT. 2, 2003

Santa Clara Convention Center

EDGE web services conference & expo

WEB SERVICES EDGE CONFERENCE & EXPO



The **Leading Magazine** for Enterprise and IT Management

LinuxWorld Magazine

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. *LinuxWorld Magazine* is aimed squarely at providing this group with the knowledge and background necessary 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. *LinuxWorld Magazine* does not feature low-level code snippets but focuses 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 presents a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features include:

Advice on Linux Infrastructure

Detailed Software Reviews

Migration Advice

Hardware Advice

CEO Guest Editorials

Recruiting/Certification Advice

Latest News That Matters

Case Studies

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY
\$49⁹⁹

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR

VISIT WWW.SYS-CON.COM



Multipass Validation with XSD and Schematron Part 2

WRITTEN BY
ERIC J. SCHWARZENBACH
& DAVID KERSHAW

Using Schematron rules to enforce good W3C schema design

In Part 1 of this article (*XML-J*, Volume 4, issue 7) we outlined why a development group might consider alternative validation schemes. An example from our experience is applying work group rules to the process of XSD design. We said rules could take the form of a Schematron schema that would be applied when a developer validates an XSD against the schema for XSD. In our past work, a need existed for a productive way to put the alternative into play without losing familiar tools or disrupting current development patterns. For that reason we developed a simple multipass validation framework using XMLSPY's Scripting Environment. In this installment, we walk you through the scripting that remains to be set up and then look at how our framework can be a productive tool for your real-world use.

Finishing the Global Functions

At this point your "(GlobalDeclarations)" should include stubs for all the functions and global variables you have seen so far. They should have the same arguments and be in a form similar to what is shown in Listing 1.

Our implementation of these functions is uncomplicated, but there are a few areas to highlight, including where we use XMLSPY's object model and our use of the Windows Scripting Host.

We chose to read and write processing instructions holding validation commands because of the relative simplicity of using Altova's XMLData object; however, since most developers are more familiar with the DOM, let's take a closer look. Altova uses the XMLData interface instead of the DOM because the internal needs of developer's tools are unique and because XMLData can easily interoperate with the DOM (you can see examples of this in the XMLSPY Code Generator examples).

We factored code for finding all PIs into a pair of functions. The lack (on the public API, at least) of a count property giving advice on the number of children of an XMLData node is notable. Until this oversight is addressed, the easiest workaround is to catch the exception XMLSPY throws at end of the list (see Listing 2; Listings 2-7 can be found at www.sys-con.com/xml/sourcec.cfm).

"getAllPI" returns the set of PIs found at or below the node passed in. To get a reference to the root element of the current document you simply walk the object model starting at "Application".

```
var doc = Application.ActiveDocument;  
var xml = doc.RootElement ;  
var array = getAllPI(xml);
```

Another tricky point is where we save the document after adding a new PI in the "writePI" method. The fragment below shows how to create and insert the PI. Then, before you can save you need to switch into Grid view. We found that Text view did not save the document and update the view reliably. After saving we switched the view back, if needed, to avoid disorienting the user.

```
var doc = Application.ActiveDocument;  
var xml = doc.RootElement ;  
var newchild = doc.CreateChild( 9 );  
newchild.Name = name;  
newchild.TextValue = value;  
xml.AppendChild( newchild );  
var mode = doc.CurrentViewMode;  
if ( mode != 0 ) doc.SwitchViewMode( 0 );  
doc.Save();  
doc.UpdateViews();  
if ( mode != 0 ) doc.SwitchViewMode( mode );
```

Now you come to the Windows Scripting Host objects. Our framework uses WSH to get the XMLSPY install directory, read and write files, and execute validation commands. There isn't anything very difficult about using WSH in XMLSPY's Scripting Environment. The most important part is just knowing what objects are available and how to get a reference to an instance of one.

We created a two-line "getSpyPath" function to return the XMLSPY install path. The files your command and schema mappings are stored in are directly under this directory.

```
var shell = new ActiveXObject("WScript.Shell");  
var value =  
shell.RegRead( "HKEY_CURRENT_USER\\Software\\Altova\\XML  
Spy\\Setup\\InstallationDirectory" );
```

Once you have a way to address the files, you need a way to read and write them. You will do this with WSH's "Scripting.FileSystemObject". We ran into a snag at this point. It's easy to get a handle to a file using WSH, but we were unable to use the "exists" method we found in online documents. Instead of testing for existence we worked around this with a conservative file create call that fails when the file exists, as shown in Listing 3.

Reading files involves an additional step. Once you have a reference to the file, use the "OpenAsTextStream" method and

iterate over the lines, as shown in the fragment below.

```
var f = fso.GetFile( doc.GetPathName() );
var s = f.OpenAsTextStream(1);
var chars = "";
while ( ! s.AtEndOfStream ) {
    chars += s.ReadLine();
}
```

Finally we get to the heart of the matter – executing validation commands. Again, the WSH call is straightforward. The “WScript.Shell” object’s “Exec” method returns a process object with a standard out stream. Every validation command must report through this stream so we can collect the output and present it to the user.

Additionally, it made sense to allow validation commands to include two delimited variables: “schema” and “doc” for the path to the schema and document files, respectively. You already created the simple form that presents the validation report. That form breaks the “validationMessage” string at every “;” character for readability. All this comes together in the lines below.

```
var shell = new ActiveXObject("WScript.Shell");
var doc = Application.ActiveDocument;
actual_command = replace( actual_command, "schema",
    actual_schema );
actual_command = replace( actual_command, "doc",
    doc.GetPathName() );
var pipe = shell.Exec( actual_command );
while( ! pipe.StdOut.AtEndOfStream ) {
    out += ";" + pipe.StdOut.ReadLine();
}
validationMessage += out;
```

Setting up the framework

There are a few quick configuration steps to take before you are done with the scripting side. First, since we had a number of small WSH issues interfacing with the file system we found it easier to create the validation command and schema text files by hand. The scripts you can download will create these files as written but they may also fail with a permissions error while in the process of writing the new file. (With more time we may be able to work around this issue, but it hardly seems worthwhile right now.) In the XMLSPY install directory, probably “c:\Program Files\Altova\xmlspy”, create “validation-cmds.txt” and “validation-schema.txt”.

Next, in XMLSPY, click on the Tools menu and select “Options”. Flip to the Scripting tab and check “Activate scripting environment when XMLSPY starts”, “Run auto-macros”, and “Process events”. Also, if you are using our scripts, check that “JavaScript” is selected. Finally, make sure that the “Global scripting project file” field points to the scripting project file containing your code. Clearly, you also need to set up utilities for your validation commands.

Now, if you’re using our project file, the next time you start XMLSPY and open a document, your multipass validation commands will appear on the Tools menu thanks to a showMacros call in the On_OpenDocument event handler. Every other member of your work group just needs a copy of the project, the command and schema files, and to configure their XMLSPY scripting environment as you just did.

That ends the hard part. The rest of the scripting work will be self-explanatory as you look at what we did. Now for the fun

part – using the framework for multipass validation.

Validation Command Code

Before you create validation rules you need a way to execute them. Our Schematron utility of choice doesn’t take much setting up. A simple but complete zvonSchematron driver could be as trivial as this .bat file:

```
c:\zvon\AltovaXSLT.exe -xml %1 -xsl c:\zvon\zvon\bin\
zvonSchematron.xsl -out c:\zvon\schema.xsl
c:\zvon\AltovaXSLT.exe -xml %2 -xsl c:\zvon\
schema.xsl -out report.htm
```

in which the arguments are a Schematron schema and the path to the document to be validated. Although this worked fine with our framework, it output more command-line feedback than we liked. Since the first step generates an XSLT file from the Schematron schema, you probably don’t want to perform that step each time you validate a file. Removing the step also eliminates some of the noise. Also, since the framework doesn’t want HTML, and we didn’t want to hack on zvonSchematron, run zvonSchematron on your schema once, and modify the resulting XSLT for simple line-by-line output. Your batch file then becomes even more basic:

```
c:\zvon\AltovaXSLT.exe -xml %1 -xsl %2 -out %3
```

At that point it makes more sense to just add that as the validation command in this form:

```
c:\zvon\AltovaXSLT.exe -xml ~doc~ -xsl ~schema~ -out
temp.txt | more temp.txt
```

of course, using your own command-line XSLT processor. Your XSLT file generated by zvonSchematron from your schema needs to be added to your collection of schemas using the framework. Both the “~schema~” and “~doc~” will be swapped out at execute time for the full path to your schema and to your current working file, respectively (see Figure 1).

Example XML Schemas

Now that we have everything ready and our macros showing on the Tools menu in XMLSPY, we need XML files to test against. As we said, the goal is to make it easy for schema developers to meet a set of workplace guidelines for XSD.

Our scenario is this: multiple developers are creating new document types from an existing framework of developed XSDs. We wish to ensure that these new schemas fit established patterns. To understand the value of the rules being enforced by the Schematron schema, we need to discuss the framework XSDs.

First, the two included schemas: the file “root.xsd” gives the base type definition to be used for defining all root-level elements (see Listing 4).

At Classwell we often create new document types. Among these types we need a certain amount of commonality. Instance documents will be processed by a single tool set and used in the same application framework (perhaps with some customization for new types, but with substantial baseline functionality). The team cannot afford rewrites for every document type.

We follow patterns that guarantee our system will continue to work with new content without modification. However, as our company grows more content, work is done simultaneously, more people create new schemas, and enforcement of pat-

Our final rule uses the context to find all non-global elements based on their occurrence somewhere deeper than directly under the schema node. If such an “xs:element” node exists, test it to see if its name is “locator” and if so report an error.

Breaking the rules

To demonstrate these rules we made another schema that breaks all five of them (see Listing 7).

With the rules in place we are ready to test them against an XSD. The steps to take are as follows:

1. Apply “zvonSchematron.xml” to “rules.schema” (our Schematron file) using Altova’s XSLT Engine (or another). We named our output file “temp.xml”.
2. Modify “temp.xml” to remove HTML formatting. You can avoid this step by removing the code in zvonSchematron that generates presentation style, but as we said earlier, we choose to keep it simple for this short article.
3. Use the “Setup Schema” macro on XMLSPY’s Tools menu to add “temp.xml” as a named “schema” available for validation within the framework (since “temp.xml” is an XSLT file this is a broad definition of schema).
4. Use the “Setup Command” macro on XMLSPY’s Tools menu to add something like: “c:\zvon\AltovaXSLT.exe -xml ~doc~ -xsl ~schema~ -out temp.txt | more temp.txt”
5. Use the “Add Validation” macro on XMLSPY’s Tools menu to add a validation step pairing the zvonSchematron command with “temp.xml” to the test document.
6. Use the “Validate” macro on XMLSPY’s Tools menu to run a test validation.

In combination with another validation command applying XMLSPY’s native XSD validator, this otherwise valid XSD is caught in five infractions of the work group rules and therefore fails to be valid in that context, despite its “technical” correctness.

Conclusion

The development example outlined here offers a practical alternative to a development team practicing ad hoc schema design guideline monitoring. Moreover, for those who need the extra descriptive power, this general approach to validation provides additional flexibility within a versatile and well-known development environment. Although XSD give the majority of XML developers more than enough power, given the complexity and scope of the XML world, exploring options like those described here is a valuable exercise for serious developers. ☛

References

- *Schema language comparison*: <http://nwalsh.com/xml2001/schematownhall/slides/foilgrp03.html>
- *Schematron*: www.ascc.net/xml/resource/schematron/schematron.html
- *ZVON*: www.zvon.org
- *Altova*: www.altova.com

AUTHOR BIOS

After working in software development in various industries over the past decade, Eric Schwarzenbach eventually came to specialize in electronic publishing. The issues of document-oriented XML are his focus: document and knowledge modelling, processing, and management frameworks, as well as XML databases and searching. He’s written custom document management systems and worked with native XML databases such as Tamino. He currently serves as the unofficial Tamino XDBA and content engineering lead at Classwell Learning Group.

David Kershaw is the professional services manager at Altova, Inc., the XMLSPY company. David brings 11 years of software engineering, project and product management to Altova. His previous positions include serving as the director of engineering at Classwell Learning Group and the group director of engineering at Organic, Inc. Mr. Kershaw received his master’s from Harvard University and his bachelor’s degree from the University of Massachusetts.

ERIC_SCHWARZENBACH@CLASSWELL.COM

DAVID.KERSHAW@ALTOVA.COM

LISTING 1.

```
// name for native XMLSPY XSD validation
var xmlspyValidation = "XMLSPY W3C Schema";
// error message
var validationError = "";
// validation message
var validationMessage = "";
//-----
// @param: name of the PI
// @param: name of the first attr
// @param: value of the first attr
// @param: name of the second attr
// @param: value of the second attr
// @returns: void
// @notes: writes a PI to the current doc
//-----
function writePI( name, att1, att1val, att2, att2val ) {}
//-----
// @param: the node name of the PI type
// @returns: array of string
//-----
function getPIValues ( name ) {}
//-----
// @param: number of the PI to remove (zero based)
// @returns: void
// @notes: will save document if successful
function removePI( number ) {}
//-----
// @param: string any PI that matches will be removed
// @notes: removes first PI that matches the text
//         passed in
//-----
function removePIByMatch( text ) {}
//-----
// @param: string command name
// @param: string the command
// @notes: adds a command to the command file
function addCommand( name, cmd ) {}
//-----
// @param: string command name
// @notes: removes a command from the command file
function removeCommand( cmd ) {}
//-----
// @returns: Array of string
// @notes: gets a list of command names
//-----
function getCommands() {}
//-----
// @param: string schema name
// @param: string full path to schema file
// @notes: adds a schema to the schema file
function addSchema( name, schema ) {}
//-----
// @param: string schema name
// @notes: removes a schema from the schema file
function removeSchema( schema ) {}
//-----
// @returns: Array of string
// @notes: gets a list of schema names
function getSchemas() {}
//-----
// @notes: validates the current document
function validate() {}
//-----
// @notes: clears and adds the validation macros,
//         (re)sets the messages to "" ('validationError',
//         'validationMessage')
//-----
function showMacros() {}
//-----
// @param: string name of the calling function
// @param: Error object
// @return: -1
// @notes: writes an error to the validation error
//         message variable
//-----
function printError( caller, ex ) {}
//-----
// @notes: opens the 'reportError' form with contents
//         of 'validationError'
//-----
function reportError() {}
```

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



REVIEWED BY CHRIS PELTZ



JavaOne 2003

Focus on XML and Web services

This year's JavaOne provided a good overview of the state of Web services today. This show report focuses on XML and Web services coverage at the event.

Java and Web Services

The J2EE community is working to better integrate Web services technologies into the platform. The J2EE 1.4 platform will provide a fully integrated Web services model, with support for SOAP 1.1, WSDL 1.1, and XML 1.0. The second beta of the J2EE 1.4 was announced during the conference.

Also announced was the availability of the Web Services Development Pack (WSDP) 1.2, which provides a set of enhanced XML and Web services capabilities. The 1.2 release included updates to the following APIs: JAXP and JAXB for working with XML; JAXM, SAAJ, and JAX-RPC for XML messaging; and JAXR for XML registries.

Additionally, the 1.2 release includes support for the new JavaServer Faces technology and XML encryption and digital signature Java APIs.

Announcements were made on the planned 2.0 releases of JAX-RPC and JAXB. JAX-RPC 2.0 will be updated to reflect the latest Web services specifications, SOAP 1.2 and WSDL 1.2. JAXB 2.0 will also be enhanced to support "partial mapping," a much-needed enhancement for XML binding. Sun also announced that JAXB, JAX-RPC, and SAAJ will be released to the open source community.

Importance of Interoperability

Increased interoperability is one of the key reasons enterprises are leveraging Web services today. However, many are still trying to determine how J2EE and .NET Web services can be integrated. The Web Services Interoperability (WS-I) organization was put in place specifically to address these interoperability issues. There was great emphasis placed on WS-I support at this year's JavaOne.

The most promising news around interoperability was the announcement that the J2EE 1.4 specification will be updated to support the WS-I basic profile (BP). While the BP addresses only the lower parts of the Web services stack, it will go a long way in resolving some of the current J2EE interoperability issues. Both the J2EE 1.4 beta 2 and WSDP 1.2 that were announced included support for the WS-I work.

WS-I is also developing sample applications demonstrating interoperability between vendors. At the JavaOne Pavilion, I was able to see live demonstrations from both Oracle and SAP demonstrating a supply chain management sample application.

Ease of Development

One of the more exciting announcements was around the J2SE 1.5 "Tiger" release. Many of the enhancements being made in 1.5 make Java development easier. This includes implementation of the Java Specification Request (JSR) 175, a metadata facility allowing methods and classes to be annotated. Building on this work is JSR 181, a BEA-led initiative that defines an annotated Java format for defining Web services in J2EE. BEA WebLogic Workshop already includes support for this through JavaDoc annotation tags.

Project Rave, a development tools platform focused on the needs of the corporate developer, will support JavaServer Faces and Web services. With this tool, a developer could construct a Web-based application and easily connect to an existing Web service through WSDL or UDDI. An early-access release will be available in the fall of 2003.

Security and Management

WS-Security and the Liberty Alliance received a lot of attention. WS-Security provides a set of SOAP enhancements to support authentication, authorization, and access control. The Liberty Alliance is an industry consortium focused on providing a framework for managing a net-

work of federated identities.

Current Java XML security initiatives are JSR105, 106, and 155, focused on XML-DSIG, XML-Encryption, and SAML. While there are no current JSRs addressing WS-Security and Liberty, vendors are already providing APIs to support these initiatives.

This year's conference also showed a growing interest in Web services management (WSM), including monitoring operational health, gaining business insights from the data, managing security, and SLAs. Challenges of WSM identified included management of distributed and heterogeneous platforms; tracking different quality-of-service levels; and managing the aggregation of Web services. An important distinction was made between managing the platform and managing the Web services themselves.

Web Services Collaboration

Web services collaboration garnered a lot of attention at JavaOne. While standards like WSCI and BP4WS have been in the press lately, the majority of the talks were focused on ebXML.

ebXML is more focused on B2B environments, providing a suite of middleware components for facilitating collaboration between trading partners. ebXML provides support for reliable messaging, trading partner agreements, registries, and business processes. One question that was raised was how ebXML related to WS-Security, WS-Reliability, and WS-BPEL.

Two JSRs specific to business processes were announced at JavaOne. JSR 207 is a specification defining metadata and runtime interfaces for incorporating business processes in Java. The Java Business Integration (JBI) JSR 208 extends J2EE to support business integration. Both JSRs mentioned the need to support the emerging standards of BP4WS, WSCI, and/or WS-Choreography. ☛

CHRIS.PELTZ@HP.COM

AUTHOR BIO

Chris Peltz is a senior software consultant in HP's Developer Resources Organization (devresources.hp.com), providing technical consulting on J2EE and Web services architectures. He brings over 10 years of software development experience in helping customers select technologies and tools for building enterprise applications.

Ektron

www.ektron.com/xmlj

Altova

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