

XML FOR THE NEXT GENERATION

XML JOURNAL

The World's Leading XML Resource

Volume: 2 Issue: 11

XML-JOURNAL.COM

FULL CONFERENCE COVERAGE



SEPT. 23-26 IN NYC

FROM THE EDITOR

De-myth-ifying XML
by Ajit Sagar pg. 5

INDUSTRY COMMENTARY

XML GUI Tools: What's the Right Model?
by Simeon Simeonov pg. 7

READER FEEDBACK

pg. 8

PRODUCT REVIEW

Epic Editor 4.2
by Arbortext
Reviewed by Kristian Cibulskis pg. 48

XML NEWS

pg. 64

BOOK REVIEW

XML Schemas: The 'New' DTDs
Reviewed by Bob Swart pg. 66

SYS-CON MEDIA

Parsing XML with Java 2,
Micro Edition



Q&A: Shedding a Little Light on XML

David Silverlight

To understand the importance of recursion in the solution, you must first understand...recursion

18

Web Services: Platform for XML Web Services

Christopher L. Miller

PART 3 Advanced Web service features in .NET add power and flexibility

22

XML Feature: DTD-Based XML Documents Using CORBA

PART 2 The static XML/Value mapping writes the code - all you do is invoke it

Jon Siegel

26

Show Report: JDJEdge/Web Services Edge 2001 Conference & Expo

The show went on in New York City

Alan Williamson

32

XML Query: XML Query and the Next Generation Web

A language that makes queries more readable



Kurt Cagle

38

XML Feature: The Brave New World of Web Services: SOAP, WSDL, and UDDI

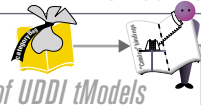
PART 2 How it all fits together

Ron Ben-Natan

42

XML in Transit: Deeper into UDDI

UDDI categorization and classification techniques and the magic of UDDI tModels



Simeon Simeonov

50

XML Transformation: Technology Option for E-Business

A jump start on the next generation of e-commerce

G. Matthew MacKenzie and Anthony Dutton

54

XSLT Tutorial: Got XSLT?

PART 2

How you can use XML and XSL transformation to create a WAP-compatible WML file



Shouki Souri

58

Iona Technologies

www.iona.com/ws-webcasts

Sonic Software

www.sonicsoftware.com

RogueWave Software

www.roguewave.com/guru2

EDITORIAL ADVISORY BOARD

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

DEPARTMENT EDITORS

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

WRITERS IN THIS ISSUE

Ron Ben-Natan, Kurt Cagle, Kristian Cibulskis,
Anthony Dutton, Jonathan Knudsen,
C. Matthew MacKenzie, Christopher L. Miller,
Ajit Sagar, Jon Siegel, David Silverlight, Simeon Simeonov,
Shouki Sourì, Bob Swart

SUBSCRIPTIONS

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

Subscription Hotline 800 513-7111

Cover Price: \$6.99/issue

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

EDITORIAL OFFICES

SYS-CON MEDIA, INC.

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

SUBSCRIBE@SYS-CON.COM

XML-JOURNAL (ISSN# 1534-9780)

is published monthly (12 times a year) by
SYS-CON Publications, Inc., 135 Chestnut Ridge Road,
Montvale, NJ 07645

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

POSTMASTER: Send address changes to:

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

©COPYRIGHT

Copyright © 2001 by SYS-CON Publications, Inc. All rights reserved.
No part of this publication may be reproduced or transmitted in any
form or by any means, electronic or mechanical, including photocopy
or any information storage and retrieval system, without written
permission. For promotional reprints, contact reprint coordinator.
SYS-CON Publications, Inc., reserves the right to revise, republish and
authorize its readers to use the articles submitted for publication.

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



**SYS-CON
MEDIA**



WRITTEN BY **AJIT SAGAR** EDITOR-IN-CHIEF 1

from
the
editor

De-myth-ifying XML

It seems that the only constant in life is change – sometimes the change is unexpected, unwanted, unwarranted. The tragic events of September 11 have left their mark. As I sat down to write this month's editorial, my mind wandered back to the way life was, and how it's changed for all of us.

At SYS-CON we felt that the best way to deal with the changes was to continue business as usual, thus defying the acts that took place a few weeks ago, acts that threatened to disrupt life in the U.S. After taking everything into account, we decided to go ahead with the XMLEdge and Web Services Edge West conference scheduled to take place this month in Santa Clara – specifically, October 22–25. By the time you get this issue, I'm sure we will have had a successful conference with participation from technologists and businesspersons who continue to contribute to initiatives in the realm of XML and Web services under all circumstances.

Not surprisingly, we have received questions about continuing with the events in the face of the tragedy. As you may know, our sister conference, JDJEdge and Web Services Edge East, took place as planned from September 23 to 26. We had a very good turnout, despite the circumstances. But both the U.S. president and the mayor of New York have urged the country to go about the business of doing business, and we concur with those sentiments.

• • •

In this editorial I'd like to address a few misconceptions and myths that are emerging around XML. The first one is that it's slow. True, XML is slower than compressed binary formats. But the key here is to identify what type of data needs to be expressed in XML. If you're transporting large chunks of data that aren't processed within a programming language environment and that deal with proprietary parsing only at both ends, there's no need to express that data in XML. XML should be used to format data that is to be shared among different applications that use a common vocabulary for expressing and interpreting the data.

Another criterion to keep in mind is that since the primary benefit XML provides over binary representations is that it is human readable, applications that don't require humans to see the data may not need to use XML. There's no "one shoe fits all" concept. For that matter, we assemble it much faster than compiled languages, but not every application requires assembly programming.

Another misconception is that XML is used only for formatting data that's transported across the Web. While this is one of the uses, there are others. For example, all the application server vendors are moving toward using XML for expressing design and runtime properties for application development and are moving away from the original properties files.

Nowadays XML is linked primarily to languages like C# and Java. However, it's also used with C++ (check out Fabio Arciniegas's book, *C++ XML*) as well as environments like CORBA for distributed computing.

Finally, due to the traction in Web services, there's a general feeling that Web services will be the only application components that XML is suited for. This isn't true either. XML finds expression in several facets of computing (e.g., MathML) that may need to be expressed as Web services. That said, XML definitely provides the building blocks for Web services.

At JDJEdge in September I met Rick Ross, president of Usermagnet and founder of JavaLobby. We had an interesting discussion on how XML and related technologies are used to address real-world issues. The case in point is the Usermagnet site <http://vigil.usermagnet.com/> that acts as a content portal for America's "Vigil on Terrorism." It's powered by XML and Java – an example of state-of-the-art technology for the latest information. Check it out. ☛

AJIT @ SYS-CON.COM

AUTHOR BIO

Ajit Sagar is the founding editor and editor-in-chief of XML-J as well as the J2EE editor of Java Developer's Journal. A senior solutions architect with VerticalNet, he is an expert in Java, Web, and XML technologies.

DataMirror

www.datamirror.com

SYS-CON MEDIA

135 Chestnut Ridge Rd., Montvale, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9607

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

ADVERTISING

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

EDITORIAL

Executive Editor
M'lou Pinkham mpinkham@sys-con.com
Managing Editor
Cheryl Van Sise cheryl@sys-con.com
Editor
Nancy Valentine nancy@sys-con.com
Associate Editors
Jamie Matusow jamie@sys-con.com
Gail Schultz gail@sys-con.com
Brenda Greene brenda@sys-con.com
Assistant Editor
Lin Goetz lin@sys-con.com

PRODUCTION

Vice President, Production
Jim Morgan jim@sys-con.com
Art Director
Alex Botero alex@sys-con.com
Assistant Art Directors
Cathryn Burak cathyb@sys-con.com
Louis F. Cuffari louis@sys-con.com
Graphic Designers
Abraham Addo abraham@sys-con.com
Richard Silverberg richards@sys-con.com
Aarathi Venkataraman aarathi@sys-con.com

SYS-CON EVENTS

VP, SYS-CON Events
Cathy Walters cathyw@sys-con.com
Conference Manager
Michael Lynch mike@sys-con.com
Sales Executives, Exhibits
Michael Pesick michael@sys-con.com
Richard Anderson richard@sys-con.com
Show Assistant
Niki Panagopoulos niki@sys-con.com
Registration Assistant
Jaclyn Redmond jaclyn@sys-con.com

JDJ STORE.COM

JDJStore Manager
Anthony D. Spitzer tony@sys-con.com

WEB SERVICES

Webmaster
Robert Diamond rob@sys-con.com
Web Designers
Purva Dave purva@sys-con.com
Stephen Kilmurray stephen@sys-con.com
Web Designer Intern
Carol Auslander carol@sys-con.com

ACCOUNTING

Chief Financial Officer
Bruce Kanner bruce@sys-con.com
Assistant Controller
Judith Calnan judith@sys-con.com
Accounts Receivable
Jan Braidech jan@sys-con.com
Accounts Payable
Joan LaRose joan@sys-con.com
Accounting Clerk
Patti Del Vecchio patti@sys-con.com

SYS-CON
MEDIA



WRITTEN BY SIMEON SIMEONOV

XMLGUI Tools: What's the Right Model?

There was a period around 1999–2000 when anything XML was hyped beyond belief. An XML-centric GUI tool, no matter how narrow in focus, attracted interest and, often enough, VC funding. The net result was a myriad of XML tools – really XML gadgets – that tried to address a large number of overlapping small problems. As a rule, all the GUI tools vied for the .xml (or .dtd, .xsd, .xsl, etc.) file extensions and interoperated poorly with other software.

Let's take a quick inventory of XML tools on my work computer: six XML document editors, two schema editors, three XSLT stylesheet editors, three data-mapping tools. And these are just the GUI tools. I have dozens more parsers, transformation, query and search engines, repositories, etc. How many of these do I use regularly as stand-alone tools? None. And that's the message I get from many people I talk to.

I believe the reason for this is that the majority of XML tools vendors miss the single most important message about XML: it's a tool (no pun intended) for getting things done, not an end unto itself. Users don't want tools that are focused on XML. They want tools that are focused on solving their problems. XML can be part of the overall picture, but it can't be the major selling point.

Another way to look at this is to ask whether working directly with XML is of primary importance for someone's job. For example, working with text is of primary importance to business writers, and that's why they use a stand-alone tool such as Microsoft Word. Software developers spend most of their days working with Java/C++/VB/other source code in integrated development environments (IDEs). Web site designers use tools such as Dreamweaver and HomeSite many hours each day. Who has to work with XML several hours per day? The main group is people building highly structured documents such as catalogs and technical manuals. They use XML in a document-centric (as opposed to data-centric) manner and need highly customizable, WYSIWYG XML instance editors such as XMetal and Arbortext.

With the exception of the people authoring highly structured documents, the rest of the professional computer users spend most of their days working in tools whose primary focus is not XML.

For example, a Web site designer may need to build a simple XSL stylesheet to transform back-end data into XHTML. A Java developer may need to modify a build script, which is an XML document. In both cases, however, people don't want to leave their primary tool of choice to get the job done. This is why Web site design tools and IDEs have become friendlier toward XML document editing. Further, in cases where the document schema is fixed (the build script being a good example), people would rather use a GUI tool specifically designed for script editing as opposed to a generic XML editor.

This environment creates a quandary for XML GUI tools vendors. They can do one of two things. They can focus on integrating with users' primary tools or they can try to secure themselves a position as a primary tool. To do the former, companies need to focus on building embeddable technologies and developing the partnerships that will allow them to deliver these technologies on top of the market leaders in the business productivity, Web design, and application development space. To do the latter, vendors will have to identify market opportunities that will lock users into their tool for some reasonable portion of their work. For example, a tool can become very successful by owning some key piece of the overall solution workflow without the need for associated infrastructure (think Visio).

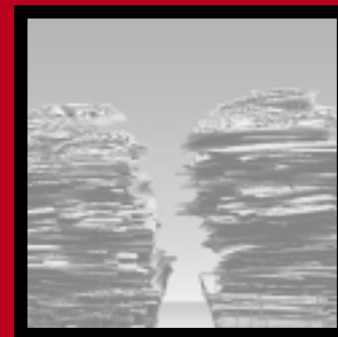
From a business perspective, the embeddable OEM model is likely to create small niche players. They'll come and go quickly as the specific problems users have to deal with change. Don't expect many IPOs in this space, but there will be some good opportunities for nimble entrepreneurs and investors. I'm still waiting for the Visio of XML tools to appear. Keep your fingers crossed. ☺

SIMEONOV @ MACROMEDIA.COM

AUTHOR BIO

Simeon Simeonov, chief architect at Macromedia, Inc., is a member of the W3C working group on XML protocol and the J2EE expert groups on XML business messaging and XML data binding.

Letters to the Editor



Online Learning Costs



As a DP manager looking for educational opportunities for a large and diverse development staff, I found Jim Milbery's article on online XML courses in *XML-Journal* [Vol. 2, issue 8] quite enlightening. I agree

with the conclusion that online courses work better for overview-type classes rather than detailed technical courses. I was wondering whether you could compare the cost of online courses for a group of about 20-25 to the cost of bringing in a teacher on-site?

Charlie Greene
via e-mail

Direct cost comparisons between on-site courses and online courses are hard to make on a purely objective basis. If you need to train geographically separated teams of developers on core technology (XML, Java, etc.), then pre-built, online courses can be very cost effective. You avoid the travel costs, and you're not paying for the cost of customized materials. Furthermore, the students don't have to take the class at the same time, which can eliminate loss time and reduce costs. The potential "soft cost" downside is that students working at their own desks may be distracted from the course work – so you're not getting your money's worth. On the flip side, a good, experienced technical instructor is going to cost you \$2,500 per day plus travel expenses. If your students are already in the same physical location, then on-site instruction should be much cheaper on a "per student" basis. In either case, I'm a firm believer in lots of lab work. Give students a little instruction and then make them get their hands dirty. In my experience, most technical students learn by "doing" as opposed to "watching."

XML-J vs WSJ

I just started getting *XML-Journal* a couple of months ago. It's a great magazine and I really like the content. However, I noticed that SYS-CON also publishes a *Web Services Journal*. How are these different? Do the contents overlap?

Ron Stewart
New York, NY



Thanks for the kudos. I'm glad you like XML-J. You've asked a good question regarding the two magazines. Several other folks have also asked this on numerous occasions. We feel there is a

need for two separate magazines to cover these technologies. If you look at the other publications in the industry, several publishers combine Web services and XML content into one publication. While there's nothing wrong with that, we feel that different business and developer groups focus on different aspects of the technologies. Having everything in a single magazine provides mixed content, all of which may not be relevant to a particular reader group.

The overlap is because Web services is a paradigm that's built on XML technologies. However, Web services are focused on a more abstract business application layer – one that exposes the functionality of applications as componentized services that can be published on the Web. XML is used to build these services. XML technologies occupy a lower echelon in the technology suite. For example,

XML is now the de facto standard for configuration files in application servers. This may not be of any interest to folks building Web services for an order management system.

XML-J does have a Web services column to focus on that particular topic. However, it also covers XML standards, XSLT, Java and XML, XML servers, XML development environments, and so on. WSJ, on the other hand, will focus on UDDI, WSDL, SOAP, and related technologies. Over time we expect Web services development to become available in application server development environments. These environments are powered by XML and related technologies.

Figuring It Out?

Interesting editorial on the different flavors of application servers [XML-J, Vol. 2, issue 10]. Do you have any recommendations on how to figure out what product to evaluate?

Florence Johnson
New Haven, CT

Glad you liked the editorial. The first thing you need to do is identify where in your application does XML play a role. Are you using it at the presentation layer for layout management? Are you using it at the database level for queries? Are you using it for passing documents in a business process?

Once you identify the areas, you need to go to the market and do the research. There are a couple of interesting sites that group XML products into specific application categories. One of the best online sites that I have seen is www.xmlsoftware.com.




Letters may be edited for grammar and clarity as well as length.

Please e-mail any comments to Ajit Sagar
ajit@sys-con.com.

Omnimark Technologies

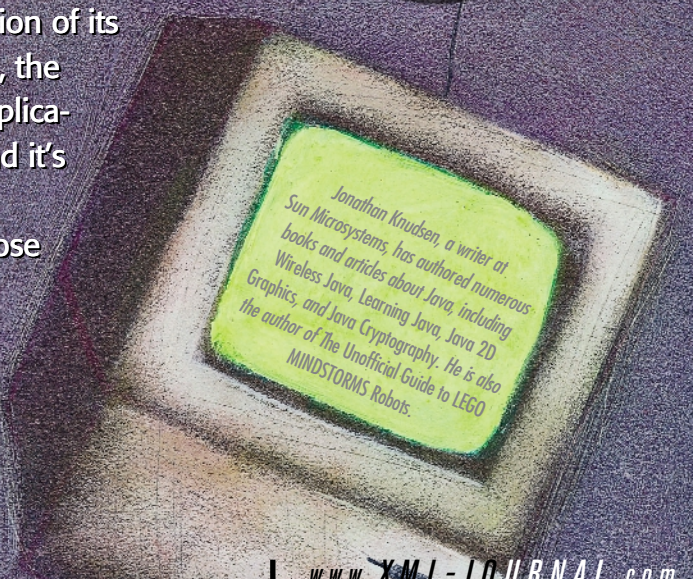
www.omnimark.com



XML GOING WIRELESS

The Java platform is undergoing the second revolution of its brief history. The first was the dramatic rise of servlets, the Java programs that are the building blocks of Web applications. Java's second revolution is in small devices...and it's happening right now.

Java 2, Micro Edition (J2ME) is Sun's name for a loose grouping of specifications that describe how to run Java on small devices, everything from mobile phones and pagers through set-top boxes and car navigation systems. Motorola, Nokia, and other device manufacturers are already producing devices that run the newest J2ME specifications.



Jonathan Knudsen, a writer at Sun Microsystems, has authored numerous books and articles about Java, including Wireless Java, Learning Java, Java 2D Graphics, and Java Cryptography. He is also the author of the Unofficial Guide to LEGO MINDSTORMS Robots.

At the same time that Java is exploding into the small-device market, XML is the hottest word on the streets since Elvis. Developers the world over are struggling to catch up with a big pile of new acronyms, everything from SAX and DOM through XSLT and FOP. XML is a language for creating data documents that are clean and portable. It's a great solution for exchanging data between applications.

J2ME is cool, and XML is cool – wouldn't it be nice if they could play together? In this article I'll walk you through the current landscape of the J2ME and XML world. It's fun stuff, right on the raw edge of technology. First we'll take a step back and talk a little more about J2ME and XML separately. After that I'll explain some of the challenges of developing on a small device. Then we'll take a brief look at existing Java APIs for working with XML. After that I'll summarize the current XML packages that are suitable for the J2ME world. I'll wrap up with some tips about how to convince XML packages to run inside a specific J2ME environment.

J2ME Overview

J2ME isn't a specific piece of software or even a specification. All it really means is Java for small devices, everything smaller than a desktop computer. This encompasses a huge disparity of devices, from pagers and mobile phones on the low end to set-top boxes and Internet appliances on the high end. The J2ME landscape, therefore, is divided into configurations and profiles.

Configurations are specifications that describe a Java Virtual Machine and some fundamental APIs for a specific class of device. A configuration specification might say something like "This configuration is designed for devices that have *x* amount of RAM and an always-on network connection. These devices will use such-and-so virtual machine and have this and that API." Currently there are two configurations, the Connected Device Configuration (CDC) and the Connected Limited Device Configuration (CLDC). The CDC, broadly speaking, is designed for large devices with reliable network connectivity. The CLDC is designed for smaller devices like mobile phones and pagers that have an intermittent network connection (hence "Limited"). It specifies the use of a small JVM, the KVM.

Profiles are based on configurations but are more explicit. They may have APIs for creating user interfaces or working with persistent storage on a device. Both the CDC and the CLDC serve as main branches of the J2ME tree, with smaller profile branches growing off the configurations. The most complete profile thus far is the Mobile Information Device Profile (MIDP), which is based on the CLDC. MIDP is designed for mobile phones and other small devices – small screens, small memory, slow processor, and limited network connectivity. In the U.S., Motorola is the first to market with MIDP-compliant devices, the i50sx and i85s mobile phones.

MIDP is built on the CLDC and includes APIs for creating applications (MIDlets), building user interfaces, and working with persistent storage. Network APIs are included with the CLDC. In this article I'll discuss how you can work with XML in a MIDP environment.

Background on XML

What's so great about XML? There's nothing really flashy, but XML has two features, cleanliness and validation, that make it very useful.

The first thing I like about XML is that it's clean. Every piece of data in an XML document is surrounded by tags. Sometimes people call this *self-describing data*. Whatever you call it, it means that XML documents are very clearly structured. The following XML document contains a word and its definition.

```
<word-information>
  <word>turgid</word>
  <definition>
    being in a state of distension : SWOLLEN, TUMID
  </definition>
</word-information>
```

If you've worked with HTML, this may look familiar. XML, however, is quite a bit cleaner than HTML. Every opening tag (e.g., `<word>`) must have a closing tag (e.g., `</word>`). XML documents are said to contain elements. The example document above contains a root element, *word-information*, as well as two child elements, *word* and *definition*. Because XML is syntactically clean, there's no confusion about which data goes in which element, or the relationship between the elements.

How do you know what names to use for elements? This is where things get really interesting. You can create a separate document, the Document Type Definition, which describes how your documents should look. In this case you could create a DTD that says something like "This type of document should have a root element called *word-information* with two child elements, one called *word* and one called *definition*." XML software can perform validation to verify that a document conforms to its DTD. (An emerging specification, XML Schema, will eventually replace DTDs, but the concept is the same.)

A *parser* is the fundamental piece of XML software that application developers will

Written by Jonathan Knudsen

Name	URL	License	Size	MIDP	Type	SAX
ASXMLP	www.alsutton.com/xmlparser	none	5 kB	yes	push	no
kXML	http://kxml.enhydra.org/	EPL	34 kB	yes	pull	no
MinML	www.wilson.co.uk/xml/minml.htm	BSD	15 kB	no	push	1.0
NanoXML	http://nanoxml.sourceforge.net/	zlib/libpng	10 kB	patch	model	no
TinyXML	www.gibbaradunn.srac.org/tiny/	GPL	13 kB	no	model	no

TABLE 1 Small XML parsers

encounter. An XML parser can read XML documents and notify your application about the contents. In the desktop computing world, parsers can perform validation to ensure that the document they are parsing conforms to its DTD or schema.

Generating XML is also important, but it's much simpler than parsing. Some parsers support generating XML and some don't.

Likelihood of Server-Side Components

The world of J2ME can be reduced to two major types of applications:

1. *Games*
2. *Enterprise application clients*

There are productivity applications, too – the standard datebook and calendar type of thing – but there's no compelling reason to use Java for these applications. Games are a popular area for Java, at least in Korea and Japan, and Java makes an attractive choice for games that are downloaded on demand. Beyond games, a powerful use of J2ME is as an enterprise application client platform. What can J2ME do as an enterprise application client? Anything you want. You can write software to control a factory floor, monitor a network, control a radio telescope.

Powerful as J2ME is, however, it's likely that you'll also end up writing server-side technology as part of a complete system. MIDP applications can access HTTP resources, so it's conceivable they could interact with an existing Web application; it's much more likely that you'll write custom server-side software to interact with the MIDP client. Web applications interact with a browser; they generate a lot of HTML that a MIDP client wouldn't care about. Furthermore, given the limited resources of a small device and currently glacial wireless network speeds, you don't want to send more data to a MIDP client than absolutely necessary.

XML Is Easily Adaptable

XML can be put to good use in a system with multiple types of clients. Suppose you have some application that should be accessible by both a Web browser and a MIDP client. A reasonable solution would be to store or generate the information from your Web application as XML documents. You could then apply a transformation to create a document that is tailored to a particular client type. One transformation might create HTML suitable for a Web browser; another would create a stripped-down XML document suitable for sending to a MIDP client.

This is one of the big wins of XML: store your data as XML, then use a transformation to create whatever format you want. It's pretty straightforward to transform an XML document into HTML, SVG, PDF, or whatever else you want, including another XML document.

XML is a useful tool, but if your MIDP client has to understand XML, you need a small XML parser.

Data Synchronization

Another area that sits squarely at the intersection of small devices and XML is data synchronization. Today's wireless data networks are pretty pathetic: typical speeds are 9.6KB and the network isn't particularly reliable. In fact, the only wireless standard everyone seems happy with these days is 802.11, dubbed "wireless Ethernet." However, it's a short-range wireless technology. (Bluetooth is also coming, but it's still a short-range system.) Between the stinky wide-area wireless networks and the short-range wireless networks, some type of synchronization is likely to be necessary. You'll probably do work on your small device, whether it's a PalmOS device, a mobile phone, or something like an iPaq, and synchronize it later. You might synchronize data to your desktop computer or to some enterprise application. It's easy to imagine an example scenario: a field technician enters data on an iPaq or PalmOS device, then synchronizes it via 802.11 when he or she returns to the office.

Where does XML fit into this picture? It makes sense to try to define an XML standard for data synchronization. If it's defined right, devices won't need to synchronize with an application-specific server and servers won't necessarily need a lot of knowledge about the devices that are synchronizing through them. The leading effort in this space is SyncML. You can read more at www.syncml.org/.

SAX and DOM

SAX and DOM are standard APIs for parsing XML. Some of the parsers listed in this article have limited SAX support. None support DOM, although some of the parsers follow DOM's method of building a document representation in memory.

The SAX API itself cannot be directly supported in MIDP because it relies on classes that aren't present. SAX 1.0 includes a `getLocale()` method that uses the `J2SE java.util.Locale` class, but this class is missing in MIDP. As we'll discuss later, there are ways of working around this.

Working in a Small Space

J2ME configurations and profiles are designed to be similar to the desktop Java (Java 2, Standard Edition, or J2SE) that you're already comfortable with. In theory, an experienced J2SE programmer can cash in on his or her Java knowledge and quickly learn the new APIs of a J2ME configuration and profile. MIDP, for example, builds on top of CLDC, which includes (among other things) subsets of the J2SE `java.lang`, `java.io`, and `java.util` packages. J2SE programmers will feel very comfortable in a MIDP environment.

**J2ME is cool, and XML is cool
– wouldn't it be nice if they
could play together?**

Cape Clear

www.capeclear.com/ca30

In practice, J2ME programming requires more finesse than just learning new APIs. Application programmers for a desktop computing environment are sometimes careless with memory and processing power. On a small device, memory is scarce, the processor is slow, and every line of code matters.

License	URL
EPL	http://kxml.enhydra.org/software/license
BSD	www.opensource.org/licenses/bsd-license.html
zlib/libpng	www.opensource.org/licenses/zlib-license.html
GPL	www.gibbaradunn.srac.org/tiny/gpl.txt

TABLE 2 Software licenses

As XML parsing, traditionally, is a relatively intensive task in terms of processing power and memory, you'll need to be careful in selecting a parser for a MIDP environment. First, you'll want to select a parser that's designed such that its code is small and it doesn't take gobs of memory to parse a document. Second, you'll have to give up on running a validating parser in a MIDP environment. Validation is fairly intense work, and the extra memory and processing requirements will reduce your mobile phone to a smoldering heap of scrap metal before you can finish parsing a document.

Don't give up on validation entirely, however – it may still be useful during your development cycle. You can use J2SE clients with validating parsers to emulate your MIDP clients. You may well flush out bugs in your XML documents this way. Once everything is running smoothly, switch over to the nonvalidating parsers in the MIDP clients.

Parser Roundup

Table 1 summarizes the existing XML parsers that are suitable for MIDP. Table 2 lists the corresponding software licenses, with links for more information.

Most of the columns in Table 1 are self-explanatory. "Size" refers to the size of the resulting MIDlet JAR if you compile just the parser source code. Keep in mind that this is a maximum; by chopping off classes and packages that you're not using, you can reduce this size in many cases. It won't reduce much for a stripped-down package like ASXMLP, but for something like kXML it's fairly easy to reduce the code size by removing packages you don't need.

A *bytecode obfuscator* can provide even more dramatic size reduction. Obfuscators were originally designed to make it hard to reverse engineer Java `bytecode` (`.class` files), but

some obfuscators also provide optimization features that will reduce the size of your code. An obfuscator uses the following techniques:

- Renames methods, variables, and classes with shorter names
- Removes classes not used by the application
- Removes methods not used by an application

The following is a short list of some of the available obfuscators:

- www.retrologic.com/retroguard-main.html
- www.alphaWorks.ibm.com/tech/JAX
- www.preemptive.com/tools/

The MIDP column indicates whether the parser compiles without modifications in a MIDP environment. While NanoXML compiles with a patch, you'll only end up with NanoXML 1.6.4, while the current version is 2.0.

The "Type" column refers to the parser's paradigm. A *push* parser works its way through a source document, spitting out events to registered listeners as it encounters tags and text in the document. This is the way the SAX API is designed. A *model* parser reads the entire document into memory and stores it in some hierarchy of objects. This is the way the DOM API works. Finally, a *pull* parser returns one piece of a document at a time; it's up to the application to keep telling the parser to get the next piece of the document.

The SAX column specifies whether the parser provides SAX 1.0 support. As I mentioned, SAX isn't MIDP compliant and any SAX parser source code will require modifications before it will build in a MIDP environment.

Porting Tips

What's involved in porting an XML parser to MIDP? Most of the problems you'll encounter stem from classes that are present in J2SE (standard desktop Java) but missing in MIDP. There are a few strategies for overcoming these problems:

- Supply dummy implementations of the missing classes.
- Use inner classes with dummy implementations.
- Remove parts of the parser.

In practical terms, you can just copy the parser's source code into your MIDP environment and try to build it. This will give you a good feel for exactly what's causing trouble. Then you can figure out how to eliminate the problems and get the parser to build in a MIDP environment. (My book, *Wireless Java*, has more specific instructions for porting and using MinML, NanoXML, and TinyXML in a MIDP environment.)

Using the Parser

Once you get a parser running in a MIDP environment, using it is generally simple. Listing 1 is a simple MIDlet that parses an XML file that's stored as a resource in the MIDlet suite JAR. It uses the ASXMLP parser.

Most of the code deals with MIDlet life-cycle and user interface. The actual parsing takes place in the `commandAction()` method. Note that, in general, it's not a good idea to perform lengthy processing inside an event handler method. If you're expecting parsing to take a long time (longer than half a second, for example), you should really perform the parsing in a separate thread so you don't lock up the user interface.

At any rate, this MIDlet tries to load a resource file from the MIDlet suite JAR as follows:

What can J2ME do as an enterprise application client? Anything you want.

Hit Software

www.hitsw.com


```
InputStream rawIn =
    this.getClass().getResourceAsStream("/example1.xml");
Reader in = new InputStreamReader(rawIn);
```

Next, we create a listener class that will be notified as the parser works its way through the document. For this simple example the listener methods are empty:

```
XMLEventListener listener = new XMLEventListener() {
    public void tagStarted(String name, Hashtable attributes) {}
    public void plaintextEncountered(String text) {}
    public void tagEnded(String name) {}
};
```

All that's left now is to create the parser and start it running:

```
XMLParser parser = new XMLParser(listener);
parser.parse(in);
```

As the parser runs through the document, the methods in the listener object are called.

Conclusion

J2ME and XML fit together well. It's an exciting time for both. The data passing between servers and J2ME clients in the future will probably be XML documents. Small parsers for J2ME devices are evolving rapidly, and there's already a nice choice of decent parsers that don't take up a lot of space. Armed with the knowledge you've gained and a little bit of source code, you're now ready to plunge into the exciting world of J2ME and XML. ☒

JONATHAN.KNUDSEN @ SUN.COM

Validation is fairly intense work, and the extra memory and processing requirements will reduce your mobile phone to a smoldering heap of scrap metal

LISTING 1

```
import java.io.*;
import java.util.Hashtable;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

import com.alsutton.xmlparser.*;

public class ASXMLPMIDlet
    extends MIDlet
    implements CommandListener {
    private Display mDisplay;
    private Form mForm;

    public ASXMLPMIDlet() {
        mDisplay = Display.getDisplay(this);

        mForm = new Form("ASXMLPMIDlet");

        mForm.addCommand(new Command("Exit", Command.EXIT, 1));
        mForm.addCommand(new Command("Do it!", Command.SCREEN,
1));

        mForm.setCommandListener(this);

        mDisplay.setCurrent(mForm);
    }

    public void startApp() {
    }

    public void commandAction(Command c, Displayable s) {
        if (c.getCommandType() == Command.EXIT) {
            notifyDestroyed();
            return;
        }

        InputStream rawIn =
```

```
this.getClass().getResourceAsStream("/example1.xml");
        Reader in = new InputStreamReader(rawIn);

        try {
            System.out.println();
            log("Creating a listener.");
            XMLEventListener listener = new XMLEventListener() {
                public void tagStarted(String name, Hashtable
attributes) {}
                public void plaintextEncountered(String text) {}
                public void tagEnded(String name) {}
            };
            log("Creating a parser.");
            XMLParser parser = new XMLParser(listener);
            long start = log("Starting to parse.");
            parser.parse(in);
            long end = log("Finished parsing.");
            long elapsedTime = end - start;
            log("Parsing time: " + elapsedTime);
        }
        catch (Exception e) {
            System.out.println("Parsing exception: " + e);
        }
    }

    private long log(String s) {
        long now = System.currentTimeMillis();
        String hexTime = Long.toString(now, 16);
        System.out.print(hexTime);
        System.out.print(" ");
        System.out.println(s);
        return now;
    }

    public void pauseApp() {}

    public void destroyApp(boolean unconditional) {
    }
}
```

DOWNLOAD THE CODE @
www.xml-journal.com

IXIASoft

www.ixiasoft.com



Recursion reminds me of...recursion

Shedding a Little Light on XML

To understand recursion you must first understand recursion. Okay, so maybe it's the oldest recursion joke in the book, but fortunately I don't have to make my livelihood as a comedian. This month's column is dedicated to questions on recursion, in both the XML DOM and XSLT.

re-cur-sion \ri-'kər-zhən\ *n* [mathematics] 1 : SEE RECURSION...

I don't often get questions that specifically ask, "How do I use recursion to...?" But I do get questions in which recursion plays a key role in the solution. The questions in this month's column are prime examples. If I were to guess why people avoid recursion, I'd say it could be because it appears to be the harder road to take. But IMHO it isn't. And at times it may be the only road.

After you see a few questions and demos, including a game of XSLT Tic-Tac-Toe (you vs XSLT recursion), I hope to change the way you view the subject. Many of the answers in this month's column can be truly appreciated only after you download and examine the source code. Here's to shedding some light on recursion...

Q: *Why does XSLT require recursion for what seem to be simple tasks? After all, many other languages can solve the same problems without using recursion.*

A: The prevalent use of recursion is a characteristic of functional languages – languages that don't have side effects like global variables. If you have a language with global variables, you don't need recursion as you can iterate through anything with a changing pointer. If you *don't* have global variables, you have to reintroduce a stack frame with a new set of localized variables with the iterated values and sets (be they sets of nodes or sets of characters).

Functionally (no pun intended), XSLT works the same magic as other functional languages. Essentially, this is because the "call-template" construct opens up a new stack frame with a new set of locally

scoped variables that can be bound with values different from the previous stack frame's locally scoped variables. This allows working with a particular item of a set and then recursively calling the same code with either a subset of the string or a different index into the set.

A forefather of XSLT, DSSSL (Document Style Semantics and Specification Language), is also a functional language and many algorithms need to be implemented recursively. XSLT is a templating language but is, in computing theory, Turing-complete, and the "call-template" construct is no different from a function call (except in its verbose syntax, which can make it awkward to write and review). If you understand the use of recursion, it's only the syntax that gets in the way, making it more difficult than the syntax of imperative languages.

Q: *How do I iterate through a string of characters in XSL?*

A: You don't (well, not exactly). For this problem the desired result is to place an HTML line break (
) before each uppercase character that's processed. The purpose of this algorithm is to iterate through a string of characters and insert an HTML line break ("
") before each uppercase character. To tackle this type of problem step by step, the first part of the solution is to create a template that will recursively process the string character by character. In fact, the bulk of the solution is contained within this template, called "ProcessString". Listing 1 shows ProcessString.xsl, the inner workings of our string processing template. Once it's coded, all that remains is to call the template, passing to it the string that needs

parsing. The following breakdown explains how to pass the string to the template and, more important, what happens within the template.

Solution

Initially, the template is called, passing each string to be parsed. In this implementation a simple "xsl:for-each" is used to feed strings contained within the "SingleString" element to the template. (Full details on ShowProcessString.xsl, the XSLT string processing template, are given in Listing 2.)

```
<xsl:template match="AllStrings">
  <xsl:for-each select="SingleString">

    <xsl:value-of select="'Original
      String: '"/>
    <xsl:value-of select="."/ ><br/>
    <xsl:value-of select="'Converted
      String: '"/>

    <!-- Call the template that will
      loop through each character of a
      string -->
    <xsl:call-template name="Iterate">
      <xsl:with-param name="strInput"
        select="."/ >
    </xsl:call-template>
    <br/><br/>
  </xsl:for-each>
</xsl:template>
```

Once the "SingleString" element is passed to the template, the remaining processing is performed within the "ProcessString" template. Listing 3, ShowProcessString.xml, provides the XML document containing the strings that will be processed. The template will process the entire string, character by character, placing a "
" before the character if it tests true as uppercase. Within each iteration the template calls itself with a new string that is one character shorter until all the characters are exhausted.

AUTHOR BIO

David Silverlight is chief XML evangelist for Infoteria Corporation (www.infoteria.com), an XML software development company based in Tokyo and Beverly, Massachusetts. He also maintains www.XMLPitstop.com, a resource for XML examples and everything else XML.

Q: What is recursion useful for?

A: As part of the source code for this article, a number of other examples demonstrating recursion are included:

- “Splitting XML Elements into Smaller XML Elements” (from Infoteria’s StylesheetCentral)
- “Transformation of Linefeed Characters to HTML” (from Mike J. Brown at Fourthought, Inc.)
- “XSLT Template to Perform Substring Replacements” (also from Mike J. Brown)
- “Trimming the Leading Spaces from a String” (from Infoteria’s StylesheetCentral)
- “Trimming the Trailing Spaces from a String” (from Infoteria’s StylesheetCentral)

The approach used in the foregoing resources is similar to that for the previous question.

Q: How can I load a tree view in Visual Basic with XML data?

A: Finally, a chance to use recursion in VB. Displaying the contents of an XML document as a tree control (with VB, XML DOM, and recursion) can be accomplished without a tremendous amount of effort. Due to the hierarchical nature of XML, it’s a prime candidate for display as a tree view. It’s also a prime candidate for recursive processing. Here’s how the three technologies combine into a single solution.

Overview

There are really two parts to loading the tree control from your XML. The algorithm is rather trivial once you see it.

Breakdown

The first part of the algorithm involves setting up the tree view. In this function the DOM is initialized, the TreeView control is reset, and the root node is extracted. In the Visual Basic application the ShowScreen-TreeView function performs the environ-

ment setup and makes the initial call to the recursive function – PopulateTreeWithChildren – passing it the root node and the TreeView control (see Listing 4).

The second part involves calling PopulateTreeWithChildren, passing the initial root node and a TreeView control as parameters. This function does all of the recursive work in processing the XML DOM object and will continue to call itself recursively while objNode.childNodes.length is greater than 1.

During each iteration, contents of the element will be added as a leaf in the tree if it’s the innermost element; otherwise it will be added as a branch of the tree (see Listing 5).

That’s about it. Once again, the recursive function in this VB application will call itself until all nodes are added to the tree. The same concepts described in earlier questions apply.

Some XML/XSL tree views worth checking out:

Fancy XML tree view: <http://skew.org/xml/stylesheetstreeview/>

Crane Softwrights Showtree: <http://www.cranesoftwrights.com/resources/showtree/>

XML TreeView in SVG: <http://www.dpawson.co.uk/xsl/sect4/d301e152>

Tree view menu implemented using XML/XSL: <http://manudea.duemetri.net/xtree/>

Q: What’s the difference between tail-end recursion and embedded recursion?

A: For starters, think of recursion in XSLT as a template that calls *itself* as a template. Questions then arise: At what point does it call itself? At the beginning of the template? Somewhere in the middle? At the end? The point at which it recursively calls itself could have serious implications on the consumption of resources. Tail end, as has been demonstrated in all of the examples thus far, includes the recursive call at the very end (the “tail end”); embedded recursion makes the call to itself anywhere within.

When determining what constitutes the “very end,” the rule of thumb is that it shouldn’t be the last call written in the

code block, but in the flow of logic the call is the last operation of the block. This means that a single block of code may have many recursive calls, all of which can be characterized as tail-end calls because each is at the end of different flows of logic within the module, not at the end of the syntax of the module.

A drawback to recursive algorithms can be the amount of stack consumed and the indeterminate amount of system resources that may be required by the processor. An XSLT processor that implements tail-end recursion will survive recursive-based algorithms far longer than an XSLT processor that does not, because tail recursion recognizes that a stack frame is no longer required during a recursive call and disposes of the old stack frame before using the new one, thus reducing the demand for stack space.

Implementation is also an important consideration when coding recursive algorithms. Saxon supports tail-end recursion while XT does not. Other processors will vary. If the coder of the routine has *any* logic after the recursive call, the implementation is obliged to keep the stack frame around and can quickly run out of stack space...hence the burden on the programmer to code carefully.

Elements of Design

Tic-Tac-Toe Using XSLT Recursion

Let’s have some fun with XSLT. This month, as a demonstration of recursion and XSLT, I’ve written a game of Tic-Tac-Toe. Here’s the catch. You’ll be using your brain to figure out your moves; the computer will be using pure XSLT recursion. Think of it as a variation on Kasparov vs Deep Blue. Are you up for it? The source code is included in this article.

Are you wondering how I did it? I started with a very common recursion algorithm (<http://erwnerve.tripod.com/tic-tactoe.htm>) and created my first proof of concept in VB6. I followed it up with a secondary version using VBScript and HTML. After the smoke cleared, and with many cups of coffee behind me, as well as a few tears...voilà: XSLT Tic-Tac-Toe. Enjoy. ☺

DSILVERLIGHT @ INFOTERIA.COM

LISTING 1 ProcessString.xsl

```
<?xml version='1.0' encoding='utf-8' ?>
<xsl:stylesheet xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template name="ProcessString">
    <!--This template will recursively process a string of
    characters and place a <br/> before each uppercase character-->

    <xsl:param name="strInput" select="''"/>

    <xsl:variable name="strCurrChar" select=
      "substring($strInput, 1, 1)"/>
```

```
<xsl:variable name="strRemainingChars" select=
  "substring($strInput, 2)"/>
```

```
<xsl:choose>
  <!--the test below will be false when strCurrChar is
  empty. -->
  <xsl:when test="$strCurrChar">

    <!-- Test to see if the character is upper case
    (in a very cool fashion),
    if so, output a <br/> -->
    <xsl:if test="not(translate($strCurrChar, 'ABCDE
    FGHIJKLMNOPQRSTUVWXYZ', ''))">
```

```
<br/>
</xsl:if>
```

<!--BTW, the test above is known as "Ken's trick" (submitted by Ken G. Holman) for a test of an uppercase character. It is impressively clever, but a bit subtle. Here's how it works:

- Translate() won't touch characters that aren't in the second argument.
- If the first argument contains only uppercase characters, they are all deleted and nothing else is touched.
- If the resulting string is empty (meaning every character was uppercase and was deleted), the empty string tests as FALSE and the not() changes it to TRUE.
- So...the entire "test=" is TRUE if the first argument contains only uppercase characters.

```
-->
```

```
<!-- Output the current character -->
<xsl:value-of select="$strCurrChar"/>
```

```
<!-- At this point, the template will recursively call
itself with the remaining
characters in the string -->
<xsl:call-template name="ProcessString">
  <xsl:with-param name="strInput"
    select="$strRemainingChars"/>
</xsl:call-template>
</xsl:when>
```

```
</xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

LISTING 2 ShowProcessString.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:include href="Iterate.xsl"/>

  <!-- Template for root rule -->
  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>

  <!-- Template for AllStrings rule -->
  <xsl:template match="AllStrings">
    <xsl:for-each select="SingleString">

      <xsl:value-of select="'Original String: '"/>
      <xsl:value-of select="."/ /><br/>
      <xsl:value-of select="'Converted String: '"/>

      <!-- Call the template that will loop through
      each character of a string -->
      <xsl:call-template name="ProcessString">
        <xsl:with-param name="strInput" select="."/>
      </xsl:call-template>
      <br/><br/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

LISTING 3 ShowProcessString.xml

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl"
href="ShowProcessString.xsl"?>
<AllStrings>
  <SingleString>OneTwoThreeFour</SingleString>
  <SingleString>MondayTuesdayWednesdayThursdayFridaySaturday
Sunday</SingleString>
  <SingleString>JanFebMarAprMayJunJulAugSeptOctNov
Dec</SingleString>
</AllStrings>
```

LISTING 4 ShowScreenTreeView function

```
Function ShowScreenTreeView() As Boolean

  Dim objXMLelement As IXMLDOMElement
  Dim tvwRoot As Node
  Dim objXMLRoot As IXMLDOMElement
  Dim objDOM As New DOMDocument
  Dim bSuccess As Boolean
  Dim strXML As String

  On Error GoTo Procedure_Err
  strgLastFunction = "ShowScreenTreeView"

  'Assume failure
  bSuccess = False
```

```
'A) Initialize and load the DOM object from the selected
  ' filename
  LoadDOMObjectFromFile objDOM, txtInputFile.Text
```

```
'B) Remove any existing elements from the TreeView
  ClearTreeView tvwData
```

```
'C) Get the root element of the XML - bypassing the
  ' comments, PI's etc
  Set objXMLRoot = objDOM.documentElement
```

```
'D) Add a child to the root node of the TreeView
  Set tvwRoot = tvwData.Nodes.Add()
  tvwRoot.Text = "<" & objXMLRoot.baseName & ">"
  tvwRoot.Expanded = True
```

```
'E) Call the PopulateTreeWithChildren function which
  will recursively drill
  ' down, starting at the root node, until all child
  ' nodes are added
  ' to the tree control
  PopulateTreeWithChildren objXMLRoot, tvwData
```

```
'If you made it to here, you have no errors
  bSuccess = True
```

```
Procedure_exit:
  ShowScreenTreeView = bSuccess
  Exit Function
```

```
Procedure_Err:
  HandleError Err
  bSuccess = False
  Resume Procedure_exit
  Resume
```

End Function

LISTING 5 PopulateTreeWithChildren() function

```
Dim objNode As IXMLDOMNode
Dim tvwChildElement As Node
Dim bSuccess As Boolean
Dim nLevel As Long
```

```
On Error GoTo Procedure_Err
strgLastFunction = "PopulateTreeWithChildren"
```

```
'Assume failure
  bSuccess = False
```

```
nLevel = tvwData.Nodes.Count
For Each objNode In objDOMNode.childNodes
```

```
'If the current node has child nodes, add the current
  'node name and text to the tree.
```

```
If objNode.childNodes.length > 1 Then
```

```
If tvwData.Nodes.Count > 1 Then
```

```
'Add a child to the tree.
```

```
Set tvwChildElement =
  tvwData.Nodes.Add(nLevel, tvwChild)
```

```
tvwChildElement.Text =
```

```
"<" & objNode.nodeName & ">"
```

```
tvwChildElement.Expanded = True
```

```
End If
```

```
'Recursively call PopulateTreeWithChildren with
  'the current node
```

```
PopulateTreeWithChildren objNode, tvwData
Else
```

```
'No more children. Add the current node
```

```
'without a recursive call
```

```
Set tvwChildElement =
```

```
tvwData.Nodes.Add(nLevel, tvwChild)
```

```
tvwChildElement.Text = objNode.xml
```

```
End If
```

```
Next
```

```
'If you made it to here, you have success
  bSuccess = True
```

```
Procedure_exit:
  PopulateTreeWithChildren = bSuccess
  Exit Function
```

```
Procedure_Err:
  HandleError Err
  bSuccess = False
  Resume Procedure_exit
  Resume
```


Altova

www.xmlspy.com

*Advanced Web service features in .NET*

A Platform for XML Web Services

Part 3 of 3

Parts 1 and 2 of this series explored the wealth of Web service–friendly features in the .NET Framework, from creating and testing Web services to building seamless proxies without typing a line of code. This final segment examines advanced techniques that provide even more power and flexibility to the XML-enabling platform.

State Management

Under many circumstances it's helpful to retain information between Web service function calls. For example, you may need to track the number of calls a particular client (consumer) makes to your service in a given period of time. While we could rely on the consumer to provide that information, we can't guarantee its accuracy. Using state management, we can persist data about our connection with a consumer for a specified period of time.

The Session object of .NET provides us with the features required to maintain state. Using this object, we'll examine a portion of code that implements state. For developers familiar with classic ASP, the Session object in .NET is practically the same for both Web pages and Web services. This object provides the properties and methods that handle session length and immediate session deletion. Session time defaults to 20 minutes, at which point it will expire and erase all stored session information.

To achieve the stateful environment, the Web server issues a SessionID that is unique. Each transaction between the client (whether a Web service consumer or simply a Web browser) and the Web server carries this SessionID in the form of a cookie. On the server side is a session store that maintains, either in memory or in a database, the individual values kept in the session object. By issuing this SessionID "primary key" to the client, we have a simple yet powerful method of retaining values with as little communication overhead as possible.

A great improvement to earlier ASP versions, session state in .NET can be maintained across multiple servers with ease. Since Web farms generally provide greater scalability, .NET session-laden

Web applications can grow well beyond the bounds of previous ASP versions.

Declaring Session State

Unlike a stateless Web service defined in .NET, the EnableSession attribute must be set to true:

```
< WebMethod(EnableSession:=True) >
```

This instructs the .NET framework to allow the passing of this SessionID information between the consumer and the endpoint.

A brief overview of our previously created CountryService Web service: we designed a Web service that would provide us some static details about a country when supplied with its two-character country code. The data returned from the Web service is then returned as XML in the browser (or brought into your application via the proxy).

Listing 1 adds stateful features to CountryService. With the addition of a new integer member to the CountryDetails class, AccessCount, we can now see the number of times we've accessed this Web service during the current session. Prior to completing each GetCountryInfo call, we add 1 to the value stored in the Session variable ("AccessCount") and place that new value in CountryDetails.AccessCount.

After requesting data from the Web service, we see the count in AccessCount rise each time we make another request (see Listing 2).

Listing 3 shows the count for a second request, and so on.

To clear the current session, issue a Session.Abandon call in your Web service. This effectively destroys the existing SessionID and all the data associated with it. If you place a Session.Abandon in the

Web service that's executed with each call, the AccessCount figure will remain at 1.

We can also set the Session.Timeout property to dictate the number of minutes a session lasts. A session is terminated after this number of minutes of inactivity. The convenience of longer sessions carries the caveat of extra memory usage. Generally, the 20-minute default is a good compromise on timing.

Modifying XML Format

Most of our .NET Web services discussion has been spent talking about creating these services and consuming them within other applications. It's also likely that you'll want to utilize the raw XML returned from Web services for transformations and other uses from time to time. The built-in Web service features of .NET allow us to customize the XML we're producing without difficult revisions to our code.

Let's first change the structure of the XML we're building from our Web service. The simplest way to arrange a level of hierarchy is to arrange our internal class structure. We'll take the three country-specific properties (population, land area, capital) and place them under the <Name> tag. Originally, our CountryDetails class was defined as in Listing 4 and in the WSDL as complexType CountryDetails. This generated the flat, one-level-deep XML we've seen in the foregoing examples. By creating a new class, CountryStats, and moving the three values into this class, we can achieve some structure to our XML, as seen in Listing 5.

The results of a call using this code look like Listing 6.

We may also prefer these values to be simply attributes of the <Name>. Adding XML attributes to our classes (see Listing 7) helps us achieve the results in Listing 8.

Once our XML is formatted to fit our requirements, we can transform this information into HTML or other presentation data with little effort.

AUTHOR BIO

Christopher L. Miller is a business consultant with Crosssoft Inc., specializing in intranet architecture and design. He is also president of the Pittsburgh .NET User Group (www.pghdotnet.org). Current projects include an adaptive intranet framework tool and other .NET-based Web service applications.

Failover to Alternate URLs

The proxy we discussed in Part 2 is a piece of autogenerated code that provides access to Web services. Proxies make the service functionality appear to reside within our application. We examined the set of properties we can use to modify the base proxy features, such as timeout and URL. Using these two properties in particular, we can build logic that allows our Web service proxy to handle timeout conditions in a clean manner.

It's important to note that URL alternatives work best with synchronous Web service calls (since timeout mechanisms are built in). In this case we've got to keep our timeout value as low as possible. In the worst-case scenario, with four alternate URLs to attempt, the client could wait the timeout number of milliseconds times five (if each were unavailable). Even a 10-second timeout would result in nearly a minute-long wait in this case.

Using a counter variable, Tries, we allow our Web service to track the number of attempts it makes to connect. With each attempt, the URL changes. We make the assumption that the parameters of the Web service don't change name or type, only location. A function SelectURL accepts the Tries variable to determine the URL to attempt to access. Listing 9 shows us the code to initialize the proxy, while Listing 10 demonstrates the code required to handle multiple URLs. We've used the names "server1", "server2", etc., to show where actual DNS or IP addresses would appear in the URLs.

Wrapping Web Services with Other Web Services

The use of Web services isn't limited to the scope of Web pages and Windows forms. In fact, a Web service may itself be a client to any number of other Web services. This is an important feature, since it presents the opportunity for "super" Web services. Imagine a corporation that manages retail stores around the globe. A single Web service could provide store-spe-

cific information to other applications throughout the organization – details like address, or phone number. Use of a Web service like this no doubt would help a company standardize information about each store. The result would be more efficient development company-wide, as well as more reliable data in each application.

A "super" Web service in this scenario (see Figure 1) could be one that provides emergency contact information for a given retail store, perhaps looking up local fire departments, police, and utilities. Requiring the store's ID as its lone parameter, this Web service could retrieve address information from the store information Web service, and then retrieve contact information from a properly licensed Yellow Pages Web service using the address details. Developers needing this emergency contact information would only need to know the store's ID to request this information.

Handling Transactions

Transactions defines a portion of logic that is executed completely or not at all. Microsoft designed the original MTS (Microsoft Transaction Server) with this in mind, and more recently incorporated and improved its features into COM, releasing COM+. Transaction support in .NET is based on this same technology.

The best illustration of a transaction-enabled function is one that involves database activity; most often, when executing a segment of database updates or deletes, it's desirable to have the transaction occur completely or not at all. After all, a partial delete or update will usually result in some loss of data integrity. With this idea in mind, Listing 11 gives us an example of a database-enabled Web service participating in a transaction.

Our example code is a Web service called RemoveExpired that deletes rows from a table. Passing it a threshold date, the SQL deletes rows whose LAST_UPDATE_DATE is on or prior to the date passed. If our Web service experiences a database connection error or something else that results

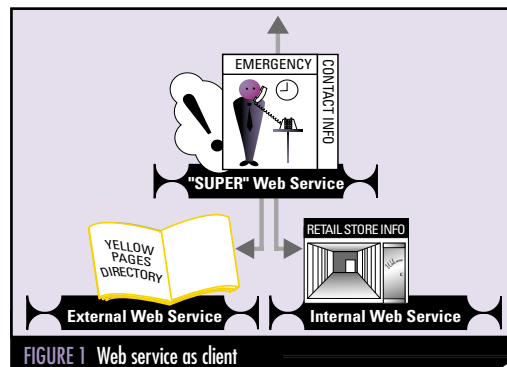


FIGURE 1 Web service as client

in a failed database transaction, we can rest assured that no rows will be deleted. On the other hand, if the Web service completes, a commit is automatically executed to ensure the entire transaction is completed.

Wrapping Up

In the three parts of this series, we've been able to examine the .NET Web service building process: how to test them, build proxies to connect seamlessly to them, and now, to add some more advanced features. In reality, this is only the beginning. The simplicity with which we have implemented each walkthrough here frees us to focus on the application of our Web services. We're only a few short months away from the full release of .NET (according to the rumors out there), so it's certainly a great time to begin learning how to take advantage of its Web services features. Even in its beta 2 maturity, we can make this assertion: when it comes to making XML Web services accessible to all, it is truly revolutionary.

References

1. *ASP.NET homepage*: www.asp.net
2. *Microsoft's .NET tutorial site*: www.getdotnet.com
3. *ASP.NET Web services information*: www.devxper.com
4. *Web services current issues*: www.webservicesarchitect.com

C MILLER @ CROSSOFT.COM

LISTING 1

Imports System.Web.Services

```
Public Class CountryService
    Inherits System.Web.Services.WebService
    <WebMethod(EnableSession:=True)> _
    Public Function getCountryInfo(ByVal strCountry As _
        String) As CountryService.CountryDetails
        strCountry = LCase(strCountry)
        Return CountryInfo(strCountry)
    End Function

    Private Function CountryInfo(ByVal strCountry As _
        String) As CountryService.CountryDetails
        Dim m_CountryInfo As New
        CountryService.CountryDetails()
        Select Case strCountry
            Case "in"
```

```
m_CountryInfo.Name = "Republic of India"
m_CountryInfo.Population = 1014004000
m_CountryInfo.Landarea = 2973190
m_CountryInfo.Capital = "New Delhi"
Case "uk"
    m_CountryInfo.Name = "United Kingdom of " & _
        " Great Britain and Northern Ireland"
    m_CountryInfo.Population = 59508000
    m_CountryInfo.Landarea = 241600
    m_CountryInfo.Capital = "London"
Case "us"
    m_CountryInfo.Name = "United States of " & _
        "America"
    m_CountryInfo.Population = 275563000
    m_CountryInfo.Landarea = 9166600
    m_CountryInfo.Capital = "Washington, DC"
Case "nl"
    m_CountryInfo.Name = "Kingdom of the " & _
        "Netherlands"
```

```

        m_CountryInfo.Population = 15892000
        m_CountryInfo.Landarea = 33920
        m_CountryInfo.Capital = "Amsterdam"
    Case "de"
        m_CountryInfo.Name = "Federal Republic " & _
            "of Germany"
        m_CountryInfo.Population = 82797000
        m_CountryInfo.Landarea = 349520
        m_CountryInfo.Capital = "Berlin"
    Case Else
        m_CountryInfo.Name = "Country not " & _
            "located in the data store."
        m_CountryInfo.Population = 0
        m_CountryInfo.Landarea = 0
        m_CountryInfo.Capital = "n/a"
    End Select
    Session("AccessCount") += 1
    m_CountryInfo.AccessCount = Session("AccessCount")
    Return m_CountryInfo
End Function

Public Class CountryDetails
    Public Name As String
    Public Population As Double
    Public Landarea As Double
    Public Capital As String
    Public AccessCount As Integer
End Class
End Class

```

LISTING 2

```

<?xml version="1.0" encoding="utf-8" ?>
<CountryDetails
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://tempuri.org/">
  <Name>Federal Republic of Germany</Name>
  <Population>82797000</Population>
  <Landarea>349520</Landarea>
  <Capital>Berlin</Capital>
  <AccessCount>1</AccessCount>
</CountryDetails>

```

LISTING 3

```

<?xml version="1.0" encoding="utf-8" ?>
<CountryDetails
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://tempuri.org/">
  <Name>Federal Republic of Germany</Name>
  <Population>82797000</Population>
  <Landarea>349520</Landarea>
  <Capital>Berlin</Capital>
  <AccessCount>2</AccessCount>
</CountryDetails>

```

LISTING 4

```

Public Class CountryDetails
    Public Name As String
    Public Population As Double
    Public Landarea As Double
    Public Capital As String
    Public AccessCount As Integer
End Class

```

LISTING 5

```

Public Class CountryDetails
    Public Name As String
    Public Stats As New CountryStats()
    Public AccessCount As Integer
End Class

```

```

Public Class CountryStats
    Public Population As Double
    Public Landarea As Double
    Public Capital As String
End Class

```

LISTING 6

```

<?xml version="1.0" encoding="utf-8" ?>
<CountryDetails
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://tempuri.org/">
  <Name>Republic of India</Name>
  <Stats>
    <Population>1014004000</Population>
    <Landarea>2973190</Landarea>
    <Capital>New Delhi</Capital>
  </Stats>
</CountryDetails>

```

```

</Stats>
<AccessCount>1</AccessCount>
</CountryDetails>

```

LISTING 7

```

Public Class CountryDetails
    <XmlElement(ElementName:="Country")> _
        Public Stats As New CountryStats()
        Public AccessCount As Integer
    End Class

Public Class CountryStats
    <XmlAttributeAttribute()> Public Name As String
    <XmlAttributeAttribute()> Public Population As Double
    <XmlAttributeAttribute()> Public Landarea As Double
    <XmlAttributeAttribute()> Public Capital As String
End Class

```

LISTING 8

```

<?xml version="1.0" encoding="utf-8" ?>
<CountryDetails
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://tempuri.org/">
  <Country Name="Federal Republic of Germany"
    Population="82797000" Landarea="349520"
    Capital="Berlin" />
  <AccessCount>1</AccessCount>
</CountryDetails>

```

LISTING 9

```

Public Class CountryService
    Inherits _
        System.Web.Services.Protocols.SoapHttpClientProtocol
    Dim Tries As Integer = 0
    <System.Diagnostics.DebuggerStepThroughAttribute()>
    Public Sub New()
        MyBase.New()
        Me.Timeout = 1
        Me.Url = _
            "http://localhost/CountryService/CountryService.asmx"
    End Sub
    ...

```

LISTING 10

```

Public Function getCountryInfo _
    (ByVal strCountry As String) As _
    CountryService.CountryDetails
    strCountry = LCase(strCountry)
    Try
        Return CountryInfo(strCountry)
    Catch
        Tries += 1
        If Tries < 4 Then
            Me.Url = SelectURL(Tries)
            Return CountryInfo(strCountry)
        Else
            Dim TempCountryDetails as _
                CountryService.CountryDetails
            TempCountryDetails.Info.Name = "Timed out."
            TempCountryDetails.Info.Population = 0
            TempCountryDetails.Info.Landarea = 0
            TempCountryDetails.Info.Capital = "n/a"
            Return TempCountryDetails
        End If
    End Try
End Function

```

```

Private Function SelectURL(ByVal Tries As Integer) As String
    Return "http://server" + Tries + _
        "/CountryService.asmx"
End Function

```

LISTING 11

```

<WebMethod(TransactionOption:=TransactionOption.RequiresNew)
> _
Public Function RemoveExpired(ExpireDate As Date) As Integer

    Dim SQL As String = "DELETE FROM RECORDS WHERE " & _
        "LAST_ACCESS_DATE <= '" & ExpireDate & "'"

    Dim Connection As New SqlConnection _
        ("user id=sa;database=rc;server=srvr")
    Dim Command As New SqlCommand(SQL, Connection)

    Command.Connection.Open()
    Return Command.ExecuteNonQuery()
End Function

```


Once you're in it...

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

reprint it...



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



Re Prints

Your Own Magazine

- Do you need to differentiate yourself from your competitors?
- Do you need to get closer to your customers and top prospects?
- Could your customer database stand a bit of improvement?
- Could your company brand and product brands benefit from a higher profile?
- Would you like to work more closely with your third-party marketing partners?
- Or would you simply like to be a magazine publisher?

SYS-CON Custom Media is a new division of SYS-CON, the world's leading publisher of Internet technology Web sites, print magazines, and journals.

SYS-CON was recently named America's fastest-growing, privately held publishing company by Inc. 500 for the second year in a row.

SYS-CON Custom Media can produce inserts, supplements, or full-scale turnkey print magazines for your company. Nothing beats your own print magazine for sheer impact on your customers' desks... and a print publication can also drive new prospects and business to your Web site. Talk to us!

We work closely with your marketing department to produce targeted, top-notch editorial and design. We can handle your distribution and database requirements, take care of all production demands, and work with your marketing partners to develop advertising revenue that can subsidize your magazine.



So contact us today!

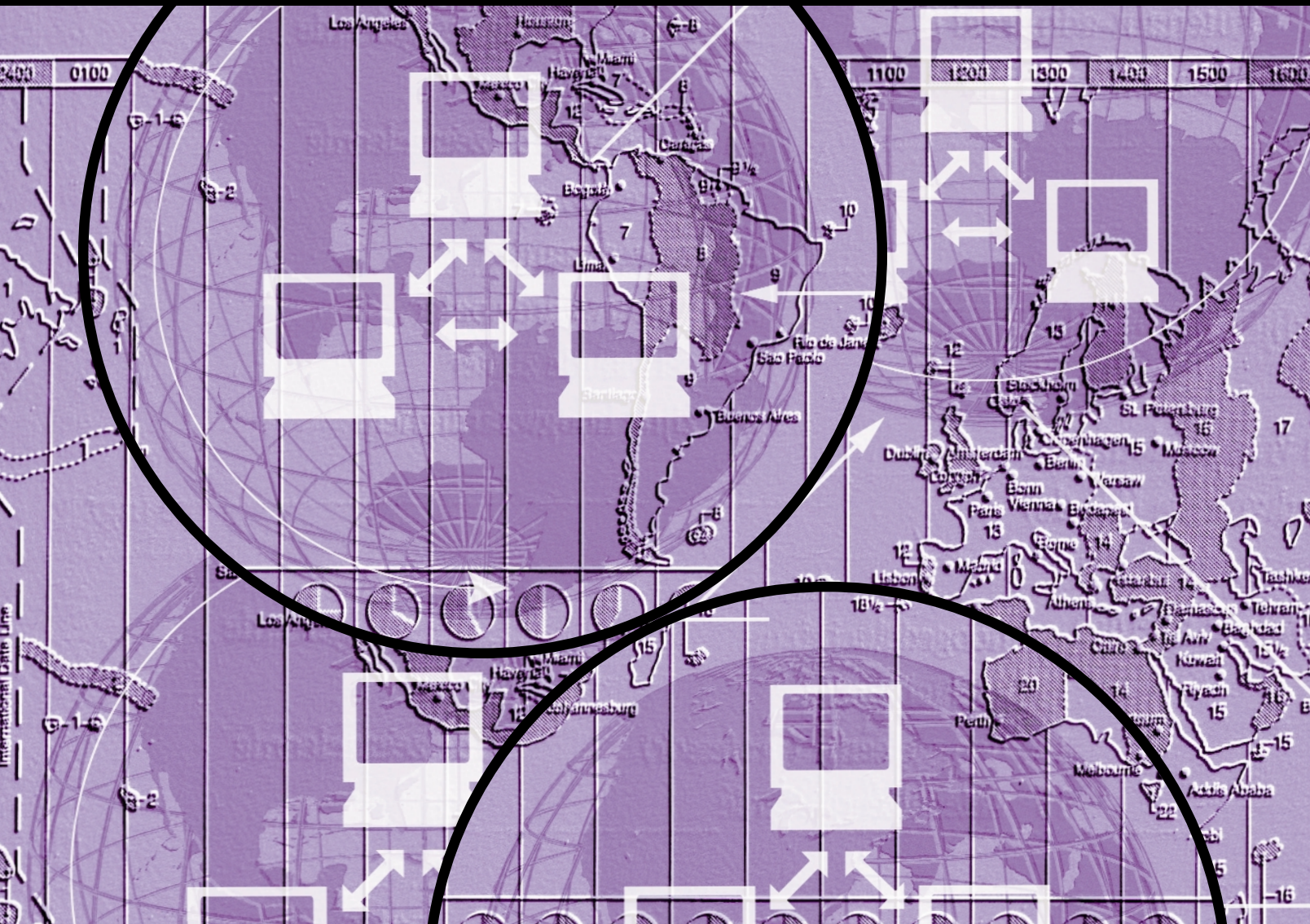
East of the Rockies
Robyn Forma
robyn@sys-con.com
Tel: 201-802-3022

West of the Rockies
Roger Strukhoff
roger@sys-con.com
Tel: 925-244-9109

X-Hive

www.x-hive.com

The static mapping writes the code – all you do is invoke it



WORKING WITH DTD-BASED XML DOCUMENTS USING CORBA'S STATIC XML/VALUE MAPPING

Written by Jon Siegel

There's a lot of XML traveling over various networks, and the interesting case is when it travels from one company to another. To make this easier to deal with and more useful, industry bodies are standardizing XML document formats as DTDs – XML Document Type Definitions – each tailored to a specific purpose.

When a company receives an XML document that conforms to a DTD, it knows what each element will be called and how the elements will be structured. (It doesn't necessarily know what to do with it, since XML is a data format and has no standard way to specify behavior, but never mind that for now.) This stability lets us create software that takes advantage of the structure to manipulate the document – write, read, edit, interpret, and ultimately act on the information contained in it. I'll refer to the part of a program that writes, reads, and changes the document as a *DTD-specific editor*, but let me point out that it's not necessarily run by a person. The information may come from anywhere. For example, a DTD-based XML purchase order (PO) could be filled in by a clerk sitting at a screen displaying a GUI, but it could be driven by an automated supply chain instead. In the automated case the PO could order hundreds or thousands of different items with no trouble at all.

In Part 1 (*XML-J*, Vol. 2, issue 10) we started our examination of OMG's XML/Value mapping with a look at dynamic XML documents – that is, those not defined by a DTD. That was just a warm-up. The dynamic mapping shows best how the mapping organizes a document and how it performs basic navigation and editing functions. In Part 2 we'll examine how OMG extends these basic functions to be DTD specific.

Using a DTD as input, the mapping not only defines a set of *valuetypes* that represent its XML elements, but also produces an implementation of these *valuetypes*. All you have to do is write the code that uses them to perform the particular manipulations that your application calls for. Since there will be many more DTD-based than dynamic XML documents traveling over the network, this is good news indeed.

In Part 2 we introduced an XML document for a simple purchase order and showed how we could read it in, edit an item, add an item, send it over the network, and write it out. We're going to use the same document again (see Listing 1), but if it looks opaque, point your browser to www.w3.org/XML to learn some XML and DTD basics. To learn about the DOM, the basis of OMG's mapping of XML to IDL, surf to www.w3.org/DOM.

DTD for the Purchase Order

A DTD defines a document's structure, separate from its content. It declares all of the element types, with their tagnames and attributes, and how they fit together. DTD structure

is flexible enough to allow some elements to contain other elements different numbers of times, but I'm not going to show how to do this here. Instead, we're going to concentrate on the CORBA mapping once we've declared our DTD. Listing 2 is the DTD for our PO.

If we were a purchasing consortium for an industry, we might standardize this to be the PO DTD that every company in the industry would use. All companies that adhered to our standards could then send POs back and forth and understand what every field means.

Every DTD, of course, defines its own set of types (if we regard each XML element definition as a type) and requires its own editor (that is, a program that reads, edits, and writes out XML documents that conform to the DTD. It doesn't matter, for our purposes, whether the input for the edits comes from a person sitting at a screen showing a GUI or is generated by a computerized supply chain – the result is the same). Industries are defining thousands of standard DTDs, so we could hire a few thousand programmers, sit them down at a few

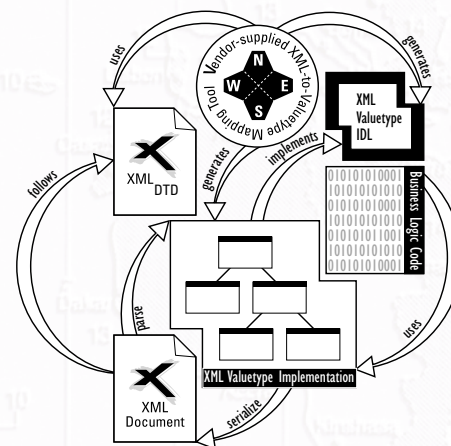


FIGURE 1 Static (DTD-based) mapping showing components and data flow

thousand keyboards, and have them program and debug a few thousand customized editors, one for each DTD, but that would be extremely inefficient. Why not generate the IDL and a set of *valuetypes* automatically from each DTD? That, it turns out, is exactly what OMG's static XML/Value mapping does. Figure 1 shows, diagrammatically, how the mapping generates the IDL and implementations for a set of *valuetypes* from a DTD.

The mapping generates two things from the information in the DTD:

1. It defines document-specific IDL interfaces for our elements, text nodes, and attributes (upper-right box in Figure 1). Instead of elements named *Element* that we can differentiate by retrieving tagnames (as we did in Part 1, using the dynamic mapping), the static mapping gives us nodes with names like *POitem* and *ship_to_address*, as we defined in the DTD for our purchase order.

Jon Siegel, PhD, is director of technology transfer at OMG, where he writes about and teaches OMG's specifications: CORBA, the CORBA services and CORBA facilities, and the modeling specifications UML, the MOF, XML, and the CWM. He is the author of CORBA 3 Fundamentals and Programming and the recently released Quick CORBA 3.



SIEGEL @ O M G . O R G

2. The mapping requires that a tool generate code for all of these DTD-defined *valuetypes* (lower-center box in Figure 1). Creation of elements with no children is easy: no new operations were defined, so all the mapping has to do is rename the *valuetype* and some of its standard operations. Creation of elements with children isn't much harder. Because the mapping uses DTD information to define parent-child relationships, document structure is built into the generated code: using DTD information to define parent-child relationships, the mapping builds document structure into the generated code. XML elements with attributes and children need interfaces to get, set, and navigate through them, so these are generated by the mapping using names assigned in the DTD. And when an element is required to have a set number of children, its create operations generate these children automatically. (For example, our *ship_to_address* must have exactly four children: one each of street, city, state, and postal code. In the generated implementation, when we create a *ship_to_address* element, it will be created with all of these children properly created, typed, and assigned.)

After the mapping tool has done its work based on the DTD, the IDL file and the document-specific *valuetype* implementations are completely coded and ready to compile. The generated implementation lays out the entire XML document for manipulation by your business logic. All you need to do is write the code to do this. The remainder of this article shows you how.

Using the Static Mapping

Following the pattern already established in this article, we'll start by working our way through the static mapping of our PO DTD and seeing what it does to our example XML PO document. Because the static mapping produces IDL, we'll look at that and not language code. Next, we'll look at code to edit an item in our PO and add a new item – the same operations we performed last month using the dynamic mapping, only easier because the static mapping gives us so much help. When we're done with our example, I'll list a few of the features in the static mapping that weren't used in our simple example.

Static Mapping for the PO DTD

This section examines the IDL that the mapping generates from our PO DTD. I've divided it into four parts, inserting explanation between them. The mapping starts by declaring our module, with a name derived from the name of our DTD. I've named the DTD and the module *purchasing*. We were careful not to name it *purchase_order*, because that's the name of our document valuetype and IDL doesn't allow names that differ only in case. Next, the IDL declares all the primary elements that we declared up front in our DTD. Listing 3 gives the ones that contain text data.

Did you notice the big difference between Listing 3 and the dynamic mapping? Here, our Elements all have type names that correspond to the names assigned in the DTD. No longer do we work with anonymous Element value-types, fetching their tagNames to see which ones we have. Instead, we operate on streets, or cities, or POitem_names!

Why truncatable inheritance? This lets you send a document over the network in a CORBA call from an application that uses the static mapping to one that uses the dynamic mapping. The standard takes care of the technical details, using some of the features of the Flyweight Pattern that also (1) save space in the in-memory representation of your document, and (2) save bits when you send it over the wire.

As we saw in the dynamic mapping, the text is held in Text Nodes that are children of their Element. Because the bottom level Elements in our PO contain only one Text Node each, operations on them are named, simply, getPCDATA and setPCDATA. If we had an Element with two Text Nodes, the operations on the first would be getPCDATA1 and setPCDATA1, and on the second, getPCDATA2 and setPCDATA2.

Now that our static IDL has declared these Text Elements, it can declare the Elements that contain them – ship_to_address and POitem. These are also DOM Elements, and each has operations to get and set the Elements that they contain. As soon as the mapping has defined POitem, it will typedef itemlist which will be used in the purchase_order (see Listing 4).

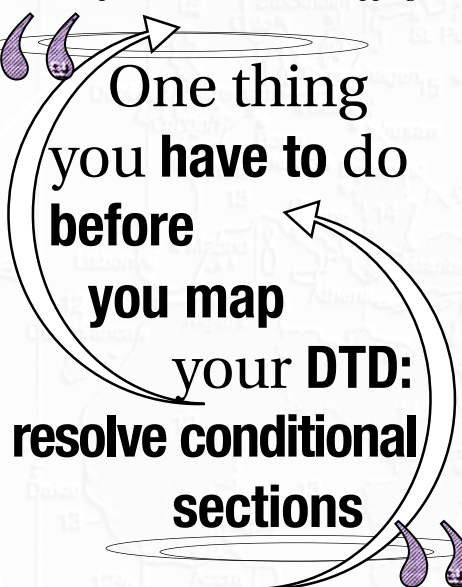
Now, finally, the mapping can declare our purchase_order. If you check back to the original DTD, you'll see that purchase_order contains a list of items. The + on the list declaration in the DTD signifies that it contains one or more, without limit. The mapping implements this as an unbounded sequence. Because the list is declared within the purchase_order in the original DTD, it is implemented as private to the purchase_order element, which contains all of the operations that manipulate list entries.

The purchase_order DTD also bore two attributes (XML attributes, not IDL attributes): company and number. These show up here too, with operations to get and set them. Listing 5 shows what the purchase_order IDL looks like.

The mapping isn't done yet. Although purchase_order holds all of our content, it's an Element and not a Document. It needs to define a Document to wrap everything up. The purchase_orderDoc bears all of the Factory operations to create elements and to start our traversal at the root. Once it has defined the purchase_orderDoc, it can declare interfaces to parse and serialize our XML document as well. And once these are declared, the mapping is indeed finished generating IDL. Listing 6 contains the last fraction of the file generated from the DTD.

Additional Static Mapping Features

The static mapping defines a few more things, mostly aspects of features we covered in the example. It defines different mappings for



the *, +, ? quantities that map to zero or more, one or more, and zero or one, respectively. It also defines mappings for sequences and choice lists (both simple and complex). And it specifies that the mapping appends numerals to duplicate names to distinguish them.

There are several pages of mappings for Element Attributes, which can contain either strings, Element IDs, references to IDs of other elements, or enumerations. I won't go into the mappings here; I'll just confirm that if you use these constructs in your DTD, it will map.

One thing you have to do before you map your DTD: resolve conditional sections. The mapping won't produce conditional IDL!

Working with the Static Mapping

I'm not going to re-present all of the dynamic mapping code with the type-specific names defined by the static mapping. The modes are too similar. But I will reprise the changing of the quantity of one of the items as well as the adding of an item. Listing 7 gives the static-mapped version of the code to change an item quantity.

Once we finish setting up our searchFor and setToQuantity variables, we start working with type names that we recognize from our DTD. Our document is a purchase_order, not just a document. And, because we have a sequence of items in our PO, the static mapping has generated operations on the sequence. (Did you notice them?) These operations are available to us, so we've used them to loop over items in search of the one we want to change. The operations have suggestive names: getPOitemSeqSize, getPOitemSeqAt(i).

Ours is a particularly simple example, with a single sequence and only two hierarchical levels of Elements that contain text. If our DTD had defined 20 times the number of element types, and our document were a hundred times the length, the identification by element name would shift from a convenience to a necessity for errorfree programming.

Listing 8 is a code fragment that creates and populates a new POitem in the static mapping.

If we were coding in the dynamic mapping, we would have had to create all of the subordinate elements – POitem_name, POitem_number, POitem_size, and POitem_quantity – individually and append them to the newPOitem as children. However, because the static mapping knows from the DTD that a POitem has exactly one child Element of each of these types, the generated code creates and appends them automatically. And each of these elements has a single Text Node child, as we'd expect. All we have to do is fill in the data.

By the way, here's an example of why all of the operations are gets instead of sets in the lines that set the values for the new item: in the first of these lines we getPOitem_name to get the POitem_name Element, then getPCDATA to get its Text Node. It's the operation data on the Text Node that's overloaded. With an input argument it's a set; without an argument it's a get and the string comes back as the return value. Here there's an input argument so this last operation in the line – data – is a set operation.

Flyweight Design Pattern

For a purchase order that has only two items, we shouldn't begrudge the few bytes taken up by the string attribute names (POitem_name, POitem_number, POitem_size, and POitem_quantity) even if the strings were stored separately as part of the state of each element. However, if our purchase order grew to hundreds or thousands of items (as it could if generated by a computer), we could save a lot of space in memory, and transmission time over the network, by eliminating the redundancy.

The XML/Value mapping does this using the Flyweight Pattern. This is an extension provided by the CORBA specification and not part of the DOM. You don't have to do any special programming to use this optimization. If you're going to send long XML documents

Missed an issue?

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

JAVA DEVELOPERS JOURNAL

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

Check out over 500 articles covering topics such as...
Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, Libraries, Embedded Java, XML, Wireless, IDEs, and much more!

JDJ The Complete Works
Reg \$119.99

Buy Online
Only \$71.99



XML JOURNAL

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

Check out over 150 articles covering topics such as...
XML in Transit, XML B2B, Java & XML, The XML Files, XML & WML, Voice XML, SYS-CON Radio, XML & XSLT, XML & XSL, XML & XHTML, 2B or Not 2B, XML Industry Insider, XML Script, <e-BizML>, XML & Business, XML Demystified, XML & E-Commerce, XML Middleware, and much more!

XML-J The Complete Works
Reg \$59.99

Buy Online
Only \$53.99



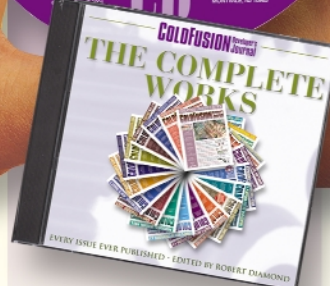
COLDFUSION Developer's Journal

The most complete library of exclusive CFDJ articles!

Check out over 250 articles covering topics such as...
Custom Tags, ColdFusion and Java, Finding a Web Host, Conference Reports, Server Stability, Site Performance, SYS-CON Radio, ColdFusion Tips and Techniques, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Application Frameworks, Error Handling, and more!

CFDJ The Complete Works
Reg \$79.99

Buy Online
Only \$71.99



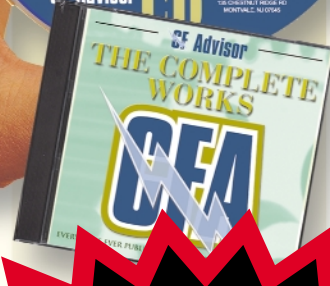
CF Advisor

The most complete library of exclusive CFA articles!

Check out over 200 articles covering topics such as...
E-Commerce, Interviews, Custom Tags, Fusebox, Editorials, Databases, News, CF & Java, CFBasics, Reviews, Scalability, Enterprise CF, CF Applications, CF Tips & Techniques, Programming Techniques, Forms, Object-Oriented CF, WDDX, Upgrading CF, Programming Tips, Wireless, Verity, Source Code, and more!

CFA The Complete Works
Reg \$79.99

Buy Online
Only \$71.99



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

JDJStore.com

Order Online and Save 10% or More!

WWW.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

with many repeated elements over the network, your network load will be much lighter if you read them into a string of valuetypes first, enable the Flyweight optimization, and send them in CORBA calls.

DOM Level 2 Features

I haven't said much about mappings for features added to the DOM in level 2, although I've mentioned support for XML Namespaces, which is the most significant of the basic manipulations we've showed so far.

The mapping also supports DOM Level 2 Events, Traversal, Range, and Views, but not features for HTML manipulation, StyleSheets, and CSS (Cascading Style Sheets).

Events may be triggered by UI devices (keyboard, mouse), logical UI events (focus change), and document mutation. Events "bubble" (in the terminology of the specification) through a target's ancestors after it is handled by the target. Or an event may be "captured" (again, in the terminology of the specification) and handled by one of the target's ancestors before it reaches the target. The interfaces may well help you build an application that edits or manipulates XML documents.

The Traversal extension defines TreeWalker, NodeIterator, and Filter interfaces for traversing DOM trees. Editors use these functions all the time, so it's helpful to have them

Events may
be triggered by
UI devices,
logical
UI events,
and document
mutation

defined in the package. It's also helpful to have a Range defined: it identifies a range of content in a Document, DocumentFragment, or Attr. Range definition supports editing, especially using a GUI: a Range is what you select from your document when you drag the mouse, for example. I'm not going to describe the IDL that supports this.

Finally, a View of a document is a computed presentation. A document may have multiple

views, depending on the stylesheet that was applied or the adjusted size of a window. Editing programs, which must identify where in a document a mouse or pointer lies, will be concerned with views. The Views interface standardizes dealing with different views.

I know this cursory treatment doesn't do justice to the capabilities of these features of the DOM. My excuse is that they're not part of the mapping of XML to CORBA per se. I admit, though, they look like a pretty good place to start if you're writing an application that presents or edits XML content.

The world has more than enough generic editors, and I don't expect many of you to start work on yet another one. Instead, I'm referring to the many content-specific XML applications as editors: the purchase-order editor we sketched out in our example is one candidate in this category. And every time an industry or organization standardizes a DTD is an opportunity for someone to run it through a Static Mapping tool, produce an IDL file and valuetype implementation, and wrap a DTD-specific editor around it.

...

XML is starting to be widely used in enterprise and business computing so I think devoting two entire articles to the discussion of XML/Value mapping was warranted. I hope you'll give this facility a try, and see for yourself how CORBA helps you manage your XML documents. ☺

LISTING 1

```
<purchase_order company="Enjay Manufacturing"
  number="01239876">
  <ship_to_address>
    <street>21 Pine Street</street>
    <city>Cleveland</city>
    <state>OH</state>
    <postcode>44113</postcode>
  </ship_to_address>
  <POitem_list>
    <POitem>
      <POitem_name>bolt</POitem_name>
      <POitem_number>BO1420</POitem_number>
      <POitem_size>1/4X20</POitem_size>
    </POitem>
    <POitem>
      <POitem_name>nut</POitem_name>
      <POitem_number>NU14</POitem_number>
      <POitem_size>1/4</POitem_size>
      <POitem_quantity>120gross</POitem_quantity>
    </POitem>
  </POitem_list>
</purchase_order>
```

LISTING 2

```
<!ELEMENT street ( #PCDATA ) >
<!ELEMENT city ( #PCDATA ) >
<!ELEMENT state ( #PCDATA ) >
<!ELEMENT postcode ( #PCDATA ) >
<!ELEMENT POitem_name ( #PCDATA ) >
<!ELEMENT POitem_number ( #PCDATA ) >
<!ELEMENT POitem_size ( #PCDATA ) >
<!ELEMENT POitem_quantity ( #PCDATA ) >

<!ELEMENT ship_to_address
```

```
( street
, city
, state
, postcode
) >

<!ELEMENT POitem
( POitem_name
, POitem_number
, POitem_size
, POitem_quantity
) >

<!ELEMENT purchase_order
( ship_to_address
, ( POitem )+
) >
```

```
<!ATTLIST purchase_order company CDATA number CDATA
#REQUIRED>
```

LISTING 3

```
#include <value_xml.idl>

module purchasing {

  valuetype street:truncatable dom::Element {
    dom::text getPCDATA ( );
    void setPCDATA (in dom::text t);
  };
  valuetype city:truncatable dom::Element {
    dom::Text getPCDATA ( );
    void setPCDATA (in dom::Text t);
  };
  valuetype state:truncatable dom::Element {
    dom::Text getPCDATA ( );
  };
}
```



```

        void setPCDATA (in dom::Text t);
};
valuetype postcode:truncatable dom::Element {
    dom::Text getPCDATA ( );
    void setPCDATA (in dom::Text t);
};
valuetype POitem_name:truncatable dom::Element {
    dom::Text getPCDATA ( );
    void setPCDATA (in dom::Text t);
};
valuetype POitem_number:truncatable dom::Element {
    dom::Text getPCDATA ( );
    void setPCDATA (in dom::Text t);
};
valuetype POitem_size:truncatable dom::Element {
    dom::Text getPCDATA ( );
    void setPCDATA (in dom::Text t);
};
valuetype POitem_quantity:truncatable dom::Element {
    dom::Text getPCDATA ( );
    void setPCDATA (in dom::Text t);
};
};

```

LISTING 4

```

valuetype ship_to_address : truncatable dom::Element {
    street      getstreet ( );
    void        setstreet ( in street Arg0 );

    city        getcity ( );
    void        setcity ( in city Arg0 );

    state       getstate ( );
    void        setstate ( in state Arg0);

    postcode    getpostcode ( );
    void        setpostcode ( in postcode Arg0);
};

valuetype POitem : truncatable dom::Element {
    POitem_name getPOitem_name ( );
    Void        setPOitem_name ( in POitem_name Arg0 );

    POitem_number getPOitem_number ( );
    Void        setPOitem_number ( in POitem_number Arg0 );

    POitem_size   getPOitem_size ( );
    Void          setPOitem_size ( in POitem_size Arg0);

    POitem_quantity getPOitem_quantity ( );
    Void            setPOitem_quantity ( in POitem_quantity Arg0);
};

```

```

typedef sequence<POitem> POitemSeq;

```

LISTING 5

```

valuetype purchase_order : truncatable dom::Element {
    //State Declaration
    private POitemSeq thePOitemSeq;

    //Attribute access operations
    dom::DOMString getcompany ( );
    void setcompany (in dom::DOMString arg0);
    dom::DOMString getnumber ( );
    void setnumber (in dom::DOMString arg0);

    //Element access operations
    ship_to_address getship_to_address ( );
    void setship_to_address ( in ship_to_address Arg0 );

    POitemSeq getPOitemSeq ( );
    void setPOitemSeq ( in POitemSeq Arg0 );
    POitem getPOitemSeqAt ( in long index );
    long getPOitemSeqSize ( );
    void replacePOitemSeqAt ( in POitem arg0,

```

```

        in long index );
void appendPOitemSeq ( in POitem arg0 );
void insertPOitemSeqAt ( in POitem arg0,
        in long index );
void removeFromPOitemSeq ( in POitem arg0 );
void removeFromPOitemSeqAt ( in long index );
void clearPOitemSeq ( );
};

```

LISTING 6

```

valuetype purchase_orderDoc : truncatable dom::Document
{
    purchase_order getpurchase_orderRoot ( );
    void setpurchase_orderRoot (
        in purchase_order docRoot );
    purchase_order createpurchase_orderElement ( );
    street createstreetElement ( );
    city createcityElement ( );
    state createstateElement ( );
    postcode createpostcodeElement ( );
    POitem_name createPOitem_nameElement ( );
    POitem_number createPOitem_numberElement ( );
    POitem_size createPOitem_sizeElement ( );
    POitem_quantity createPOitem_quantityElement ( );
    POitem createPOitemElement();
};

local interface purchase_orderParser:XMLValue::XMLParser
{
    purchase_orderDoc parsepurchase_order
        ( in XMLValue::DOMString XMLStream)
        raises (XMLValue::XMLException);
};

local interface purchase_orderSerializer:
    XMLValue::XMLSerializer
{
    dom::DOMString serializepurchase_order
        ( in purchase_orderDoc XMLStream)
        raises (XMLValue::XMLException);
};

```

LISTING 7

```

// Modify any POitem with POitem_number "B01420" to
// have a POitem_quantity of "150gross"

DOMString searchFor = makeDOMString("B01420");
DOMString setToQuantity = makeDOMString("150gross");

purchase_order order = PO_doc.getpurchase_orderRoot();
for (int i = 0; i < order.getPOitemSeqSize(); ++i)
{
    POitem thePOitem = order.getPOitemSeqAt(i);
    POitem_number number = thePOitem.getPOitem_number();
    if (number.getPCDATA().data()._equals(searchFor))
    {
        POitem_quantity quantity =
            thePOitem.getPOitem_quantity();
        quantity.getPCDATA().data(setToQuantity);
    }
}

```

LISTING 8

```

// Create a new POitem
//
POitem newPOitem = PO_doc.createPOitemElement();
newPOitem.getPOitem_name().getPCDATA().
    data(makeDOMString("nail"));
newPOitem.getPOitem_number().getPCDATA().
    data(makeDOMString("NL4590"));
newPOitem.getPOitem_size().getPCDATA().
    data(makeDOMString("1/4"));
newPOitem.getPOitem_quantity().getPCDATA().
    data(makeDOMString("200gross"));

```

★ *The Show Goes On* ★

written by Alan Williamson



THE SOFTWARE INDUSTRY RETURNED TO NEW YORK CITY WITH THE INTERNATIONAL JAVA & WEB SERVICES CONFERENCE AND EXPO AT THE HILTON NEW YORK.

Web Services Edge 2001 East International
Web Services Conference & Expo was colocated
with the JDJEdge 2001 International Java
Developer Conference & Expo

New York, NY, September 23, 2001 – In spite of what many thought might prove insurmountable obstacles, the international software industry has provided New York City today with a resounding indication that heavy hearts and thoughts are not to be permitted to become a barrier to returning to the business of business, including the Internet technology business.

Delegates from various parts of the country and from around the world began gathering at the Hilton New York to attend the leading Java and Web services technology events on the East Coast this year, JDJEdge 2001 International Java Developer Conference & Expo and Web Services Edge 2001 International Web Services Conference & Expo East, both produced by SYS-CON Events, Inc. www.sys-con.com.

Coming so soon after the devastating World Trade Center carnage, this is a strong sign that America's software developers and vendors alike are determined to go forward – coming together – to learn, to network, and to do business with each other. The first conference sessions were well attended, in one case so well that there was standing room only.

Between September 23 and 26, **SYS-CON Events** hosted its **JDJ/Web Services East** conference in the heart of Manhattan. In light of the tragic events a couple of weeks earlier, we made the agonizing decision to follow the ad-



David Litwack CEO of SilverStream

vice of Mayor Giuliani and get back to normal as quickly as possible. With that, we had near full attendance from our exhibitors and a nominal no-show from some of our speakers, who understandably felt safer not flying.



Fuat Kircaali (Founder and CEO SYS-CON Media), Tyler Jewell (BEA Systems), Dave Chappell (Sonic Software) and Dr. Richard Mark Soley (OMG) at the Expo Floor.

Web Services Journal editor-in-chief Sean Rhody opened up the conference to a packed hall and indicated the value that attendees would be getting in the next few days. This included a rich overview of Java

and the prolific implementation and adoption of Web services. James Gosling presented the first keynote, offering insight into Java and where it was heading with respect to the whole Web services revolution.

Gosling delivered a small anecdote illustrating the need to embrace and open up our technologies for greater interoperability by drawing an analogy to the airline industry after it moved to jet turbine engines. In the old days of aviation, you navigated your plane by simply hanging your head out the window, looking for identifying landmarks on the ground. With the advent of jets, the logistics of sticking your head out the window became rather more life threatening. A new breed of pilots had to rely on technology for their bearings, but now at least they could complete their journey 10 times faster.

The computer industry, Gosling continued, was going through the same change: "Just think of the applications that would be possible if we were to stop putting our heads outside and work together."

This was the main theme of the conference that carried through to both the sessions and the exhibitors. Walking around the show floor I got the feeling that change was in the air. When I stood back and looked at several exhibitors' booths, it was interesting to see that the ability to communicate to outside systems was the top selling point. This was comforting to see, and I believe a sign of an industry maturing and starting to get down to the business of delivering on the promises of the evangelists.

Speaking of evangelists, there was no shortage of them at **JDJEdge**. Technology evangelists are a funny breed, and from my experience they generally fall into one



ICES EDGE 2001

web services **EDGE**TM
conference & expo



James Gosling, Ajit Sagar (XMLEdge 2001 conference tech-chair and editor-in-chief of *XML-J*) and Alan Williamson (editor-in-chief of *JDJ*) at the Web Services Edge 2001 East Expo Floor.



Dr. Richard Mark Soley, Chairman and CEO of Object Management Group, delivering his keynote speech at the Web Services Edge 2001 East.

of two camps: those that can, and those that cannot. I've found that the ones that have captured my interest (and the room in which they speak) are those that have a coding background and still consider themselves developers. The likes of Dave Chappell from Sonic and Tyler Jewell from BEA are two great examples. They animated their sessions with thoroughly insightful and enjoyable technical discussions on Java as a whole without continually plugging their respective company's products. They inspired their audiences to go and build more open systems.

The second day opened up with a Web services panel of the industry's heavyweights, chaired by our own Sean Rhody. Don Leclair, James Gosling,

Richard Soley, Tyler Jewell, Dave Chappell, Rick Ross, and Dave Litwack answered questions from both Sean and the floor. It was interesting to note that the panel spent a significant amount of time defining exactly what the term *Web services* connotes.

"I applaud the 'show must go on' attitude, both on the part of the attendees and SYS-CON...I am very impressed with the high caliber and technical adeptness of the audience"

—Dave Chappell,
VP and SonicMQ Chief Technology Evangelist

Richard Soley argued the marketing-hype card and that we would all be talking about something else in two years' time. It just goes to show the infancy of this emerging market, that the technical definition of a *Web service* is hard to nail down.



Over 1,200 delegates welcomed James Gosling's opening keynote on Monday, September 24.



CONFERENCE & EXPO 2001

JDJEDGETM
conference & expo

One thing the panel did agree on was the need for more open systems. Whether or not XML was the best tool for the job came up as well, with James Gosling and Richard Soley both playing the performance card – there is too much redundancy inherent in the protocol, which only serves to eat up valuable bandwidth and unnecessary parsing. SavaJe announced the winner of its developer's competition that was run in conjunction with *Java Developer's*

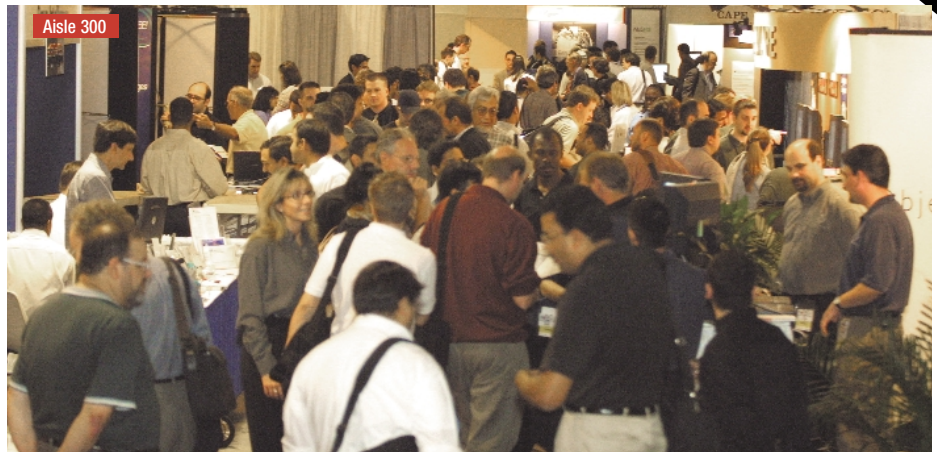
Journal. The winner of the large LCD monitor was Passport for its Java Remote Presentation Applet. Full details on the winners can be found at www.savaje.com.

On the whole the show was well attended and was just the perfect size to ensure that delegates were able to see everything they needed to without rushing around like headless chickens as they do at JavaOne! ☺

web
services **EDGE**TM
conference & expo



Exhibitors made numerous contacts at the Expo.



The largest gathering of Web services professionals in the industry.

KEYNOTE PANEL: WEB SERVICES PARADIGM



Web Services Keynote Panel: (left to right) Don Leclair, Computer Associates; James Gosling, Sun Microsystems; Richard Soley, OMG; Tyler Jewell, BEA Systems; Sean Rhody, *Web Services Journal*; Dave Chappell, Sonic Software; Rick Ross, JavaLobby; David Litwack, SilverStream Software.



Watch Web Services Edge 2001 on
SYS-CON Television at www.sys-con.com



2002

JUNE 24-27 JACOB JAVITS CONVENTION CENTER NEW YORK, NY

JAVA, XML AND .NET
TECHNOLOGIES



James Gosling,
The Father of Java



web services **EDGE**
conference & expo

INTERNATIONAL WEB SERVICES CONFERENCE & EXPO

JDJ **EDGE**
conference & expo

INTERNATIONAL JAVA DEVELOPER CONFERENCE & EXPO

XML **EDGE**
conference & expo 2001

INTERNATIONAL XML CONFERENCE & EXPO

Fundamentally Improving the Speed,
Cost & Flexibility of Business Applications

Who Should Exhibit...

Java, XML, Web services and .NET Technology vendors,
staking their claim to this fast-evolving marketplace.

JDJEdge 2002 and Web Services Edge East 2002 Will Feature...

- Unmatched Keynotes and Faculty
- Over 150 Intensive Sessions and Fast Tracks
- The Largest Independent Web Services, Java and XML Expos
- An Unparalleled Opportunity to Network with over 5,000 i-technology professionals

Who Should Attend...

- Developers, Programmers, Engineers
- i-Technology Professionals
- Senior Business Management
- Senior IT/IS Management
- Analysts, Consultants

THE CALL FOR PAPERS

WILL OPEN ON NOV. 30, 2001

VISIT WWW.SYS-CON.COM FOR DETAILS

FOR MORE INFORMATION

SYS-CON EVENTS, INC.
135 CHESTNUT RIDGE ROAD
MONTVALE, NJ 07645

201-802-3069

WWW.SYS-CON.COM

**SYS-CON
MEDIA**
OWNED & PRODUCED BY
**SYS-CON
EVENTS**

JDJ STORE.COM GUARANTEED! LOWEST PRICES!

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

Terms of offer:


- Offer good through December 31, 2001
- Only applicable to pricing on current versions of software
- Offer does not apply toward errors in competitors' printed prices
- Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

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

SYS-CON MEDIA

**XML-Journal:
The Complete
Library**



JDJStore.com **\$53⁹⁹**

SYS-CON MEDIA

**JDJ, CFDJ, XML-J
The Complete Works**


Collect All Three!



JDJStore.com **\$175⁹⁹**

SYS-CON MEDIA


**CFDJ:
The Complete
Library**



JDJStore.com **\$71⁹⁹**

MACROMEDIA®

HomeSite 4.5




HomeSite™ 4.5 from Macromedia (HomeSite) is the award-winning HTML editing tool that lets you build great Web sites in less time, while maintaining pure HTML. Created by Web developers for Web developers, only HomeSite gives you precise layout control and total design flexibility, while delivering the latest Web technologies.

HomeSite 4.5 **\$93⁹⁹**

BORLAND

Borland Delphi 6 Enterprise




Delphi 6 makes next-generation eBusiness development with Web Services a snap. BizSnap Web Services development platform simplifies business-to-business integration by easily creating Web Services. DataSnap Web Service-enabled middleware data access solutions integrate with any business application. Rapidly respond to e-Business Web presence opportunities ahead of the competition with WebSnap, Delphi's complete Web application development platform.

Borland Delphi 6 Enterprise **\$2949⁹⁹**

SYBASE

**Adaptive Server Enterprise
for WINNT v12.0**



Sybase Adaptive Server Enterprise 12.0 is designed to support the demanding requirements of Internet and traditional, mission-critical OLTP and DSS applications. The efficient multithreaded architecture, internal parallelism and efficient query optimization of Adaptive Server Enterprise delivers unsurpassed levels of performance and scalability.

Adaptive Server Enterprise WINNT v12.0 **\$798⁹⁹**

POLYCOM

**VoiceStation Personal
Teleconferencer**



Designed to improve your business communications while increasing productivity, VoiceStation 100 enables you to conduct remote meetings that are as natural as speaking face-to-face. Features: Full-Duplex Audio Performance featuring Polycom's Acoustic Clarity Technology™, Dynamically Adjusts to Any Small Room's Acoustic Environment, 3 Microphones w/ 360o Room Coverage, High-Quality Speaker, Easy to Install / Easy to Use, Redial, Flash, Hold & Mute, Data Port (110V versions only), RCA Output Jack.

VoiceStation Personal Teleconferencer **\$279⁹⁹**

CASIO

EM500 Multimedia Cassiopeia



The new slim-designed EM-500 is a mobile multimedia tool that targets a younger market. Designed to highlight Casio's new faster processor, and available in five different colors, the EM-500 is stylishly designed and engineered to take advantage of the new and emerging digital content that is available on the Internet. (For online browsing and e-mail, an optional modem is required and sold separately.)

EM500 Multimedia Cassiopeia **\$428⁹⁹**

eHELP

RoboHELP Office 2000



RoboHELP Office provides a user-friendly WYSIWYG authoring environment for creating JavaHelp. RoboHELP guides you through the process so you can create a great JavaHelp system with point-and-click and drag-and-drop ease. Now you can create JavaHelp systems as easily as you create WinHelp, Microsoft HTML Help and WebHelp (cross-platform Help) from the same source product – all with RoboHELP Office.

RoboHELP Office 2000 **\$898⁹⁹**

**BUY THOUSANDS
OF PRODUCTS AT
GUARANTEED
LOWEST PRICES!**



**GUARANTEED
BEST PRICES
FOR ALL YOUR
JAVA-RELATED
SOFTWARE
NEEDS**

SYBASE **SQL Anywhere Studio v7.0**

Sybase® SQL Anywhere Studio is a comprehensive package that provides data management and enterprise synchronization to enable the rapid development and deployment of distributed e-business solutions. Optimized for workgroups, laptops, handheld devices and intelligent appliances, SQL Anywhere extends the reach of a corporation's e-business information to anywhere business transactions occur.



SQL Anywhere Studio (base w/ 1 user) v7.0 \$348⁹⁹

MICROTEK LAB **Scanmaker 4600 with MN100 Camera Bundle**

Internet ready scanner for home or office. 3 Easy Touch Buttons for scan, copy and email provide real convenience. With 2400x1200dpi and true 42-bit color, this scanner provides razor sharp detail and richly saturated colors. Combination digital camera/PC camera included.

Specifications:
Interface Type USB, Platform Windows and Macintosh, Included Hardware USB Cable, Power Adapter, camera.



Scanmaker 4600 w/ MN100 Camera Bundle . . . \$164⁹⁹

JASC **Quickview Plus v6.0**

Now you CAN open virtually any file and email attachment with Quick View Plus - the easy way to view virtually any file. Saves money and time - views files from programs you don't have installed on your computer. With just a mouse click, Quick View Plus gives you instant access to over 200 Windows, DOS, Macintosh, and Internet file types. That's more file formats than any other file viewing utility!



Quickview Plus v6.0 \$45⁹⁹

SMC **Wireless 11 Mbps Access Point with PC Card**

The wireless LAN solution is the easiest alternative to a traditional wired network. It connects instantly with an existing Ethernet installation to support mobile users, temporary work sites, and other applications ranging from the classroom to the boardroom. Wireless technology provides the maximum user mobility, simple and flexible installation options, a reduced cost of ownership (no cabling costs or maintenance) and excellent scalability in supporting network growth. True 11 Mbps wireless is ideal for use with cable modems, DSL or SOHO applications.



Wireless 11 MBPS Access Point \$339⁹⁹

SITRAKA **JClass Enterprise Suite v 4.5.1 Bytecode/Gold Support**

Fully scalable to mission-critical development environments, JClass Enterprise Suite is the world's leading collection of Java components. JClass offers unrivaled IDE, JDK and platform support, and includes one year of Gold Support with free upgrades. 100% Pure JavaBeans and JDK 1.1, Swing/Java2 & JDK 1.3 support with 4.5.13.6 version still available Works in your IDE and supports the latest JDK; leverage advantage of Java 2 and Swing (PLAF, drag-and-drop, etc.3.6 supports earlier versions of JDK.



JClass Enterprise Suite v 4.5.1 \$3100⁹⁹

INTERLINK **RemotePoint RF Wireless Handheld w/ Software**

No software required for mouse and laser functions!
-Omni-directional control from up to 100' away!
-Dedicated Slide Forward & Back buttons -
Presentation Software Effects -Launch programs -
Zoom or spotlight images on the screen -Hide & reveal slides -On-screen keyboard -Many more!



RemotePoint RF Wireless handheld w/ Software \$169⁹⁹

MACROMEDIA® **JRun Server 3.0 Enterprise 2 CPU Licenses**

JRun™ 3.0 is an easy-to-use J2EE application server and integrated development environment for building and deploying server-side Java applications. From e-commerce to business automation, JRun is the easiest way for developers to deliver advanced business systems faster and at a lower cost than you'd ever thought possible.



JRun Server 3.0 Enterprise (2 CPU Licenses) . . . \$8602⁹⁹

LINKSYS **WAP11 Wireless Network Access Point**

The Instant Wireless Network Access Point from Linksys delivers the freedom to configure your network your way. Utilization of "state-of-the-art" wireless technology gives you the ability to set up workstations in ways you never thought possible; no cables to install means less expense and less hassle. The Instant Wireless Access Point's high-powered antenna offers a range of operation of up to 800 feet, providing seamless roaming throughout your wireless LAN infrastructure; an advanced user authentication feature ensures a high level of network security.



WAP11 Wireless Network Access Point \$174⁹⁹

SITRAKA **JClass Chart 4.5.1 with Gold Support including Bytecode**

JClass Chart gives you the power to create sophisticated, interactive graphs and charts quickly and easily. This data-aware Java component supports many popular types of business and scientific charts, which can be populated with data from a variety of sources including XML.



JClass Chart 4.5.1 Bytecode/ Gold Support \$1299⁹⁹

POINTBASE **Mobile Edition 3.5**

PointBase Mobile Edition is a powerful, 100% Pure Java™ object-relational database specifically designed for mobile client applications, Internet appliances and wireless devices. PointBase's revolutionary distributed data management delivers the comprehensive capabilities needed to enable these systems, including ultra-small footprint, extensibility, integration with enterprise databases and an entirely new level of self-management and usability features.



PointBase Mobile Edition 3.5 \$194⁹⁹

ADAPTEC **SCSI RAID 2100S PC Card**

The Adaptec SCSI RAID 2100S card delivers high performance and availability for entry-level servers - conventional and 1U/2U rack-mount - and workstations. It offers an affordable, full-featured, half-sized, micro-processor-based Ultra160 SCSI RAID solution for uses that require greater uptime, high-speed throughput, scalability, and space saving features.



SCSI RAID 2100S PC Card \$404⁹⁹

ARCHOS **Portable 6GB MP3 Player/USB**

Enjoy listening to over 100 hours/6000 minutes of CD-quality music with your ARCHOS Jukebox 6000 MP3 Player/USB Hard Drive. You can conveniently store all your personal files, along with your favorite music selections, and listen to music for 8 hours before recharging the 4 NiMH batteries. Package Includes: Jukebox 6000, 4 Rechargeable NiMH Batteries, MusicMatch Software, USB Interface, A/C Adapter, Stereo Headphones and Pouch.



Portable 6GB MP3 Player/USB \$268⁹⁹



A language that makes queries more readable

XMLQuery and the Next Generation Web

Anyone who has ever done a search query on the Internet is familiar with the phenomenon in which a single query pulls up more than a million possible search matches. This has to do with the fact that information is ultimately not linear, but rather is linked and interrelated in ways that can't be quantified easily through text searches.

When I was in college at the University of Illinois (way too many years ago to want to dwell on here), I spent an inordinate amount of time in the library – sometimes researching things for class, more often just researching things for my own interest. The library there was something of a marvel, with row upon row of cards in their card catalog, designed in such a way that you could – if you knew the author or title of a book – determine where it was in the million-plus book stacks.

Of course, this works only when the books have titles that correspond to the information you're looking for, and even then only if the first word of the title has sufficiently relevant information. Thus (accounting for the dropping of obvious keywords) *A Survey of Theory on Musical Trends* and *Survey Techniques for Municipalities* would be in the same general proximity, even though they have remarkably little in common.

While the stacks are still there as part of the atmosphere of the library, they're now years out of date. In their stead, computers with browser interfaces communicate with a large centralized database capable of storing relevant abstracts and keywords, information that was in the cards in the first place but couldn't be indexed easily. However, what has happened is that one method of indexing that probably missed too much information has been replaced by another that generates too much information.

The Web is inexorably moving to an XML basis, though perhaps not in ways that anyone would have predicted in 1996 when XML was born. Back then, the vision seemed to be that content

would exist in XML documents that could be referenced through a series of pointers that included a way of navigating to specific elements. Xpointer still exists, but it's been adopted only marginally, in great part because a significant amount of the XML that now exists on the Web comes not from static documents but rather from large databases or autogenerated server applications.

XPath, on the other hand, has become an integral part of XSLT (though not, with the exception of the Microsoft parser, DOM). The reason has to do with the need to be able to retrieve sets of nodes from a given XML document. XSLT in fact is a very simple language by itself – it provides a logical structure to tie together matches, defines variables and parameter methods, and handles the generation of output. However, without the ability that XPath brings – either to generate or to manipulate sets of nodes – XSLT would be virtually worthless.

Unfortunately, however, XPath itself has some serious limitations. It is fundamentally document oriented. A path in general extends from a single root node, can become fairly complex to write, and by itself can't retain state information – it's dependent on XSLT to do that. Also, in the current implementation, you can't use XSLT to generate functions that can be referenced from within XPath, which means that XSLT implementations of complex functions can become extremely verbose. Finally, XPath can get fairly cryptic, especially when an XPath expression has to reference an item indirectly through an IDREF or similar construct.

Introducing XML Query

The database world faced a similar problem in the mid-1980s. Relational databases had been slowly but steadily replacing flat databases for some time, but every database company had its own set of commands and terminology to access the database. This in turn meant that you had to educate your programmers every time you changed databases, and it also meant that the only people who could access your database directly were the programmers.

After some wary circling on the parts of the various combatants...er, participants, the SQL standard finally emerged as a common format for performing database operations, regardless of the vendor. While considerable variation exists among the various implementations of SQL, these variations are still relatively minor compared to having to learn a completely different programming language for each database you support. The initial version, released as an ANSI standard in 1987, covered most of the basic operations for both manipulating data and creating the corresponding data structures. SQL-89 included several fixes that made the model more robust after two years of experimentation.

The requirements for XML Query are roughly analogous to those that resulted in the later SQL-89 specification, with a typical XML twist. One of the primary issues that many people who work with XPath have with the language is that it can often be incredibly terse and cryptic, requiring an intimate association with the data that, while powerful, is often confusing. For instance, suppose you have two XML documents, one giving authors' names and the ISBN numbers for each book they've published, the second consisting of book titles with their associated ISBN numbers. You want to

AUTHOR BIO

Kurt Cagle, a writer and consultant specializing in XML technologies, is currently working on a book about programming the Semantic Web. Most recently he coauthored *Professional XSL* (Wrox). Kurt can be reached at www.kurtcagle.net or via e-mail.

produce a list showing the titles of a given set of books by a specific author. With XPath, to get the set of such books can require a fairly cryptic expression:

```
<xsl:variable name="bookTitles" select=
"document('books.xml')//book[isbn =
(document('authors.xml')//author[.='Kurt
Cagle']/isbn)]/title"/>
```

While such an expression is reasonably compact, it is, to put it mildly, more than a little hard to follow (especially for people who've spent a lot of time working with SQL-like expressions). XML Query is intended to make such a query more readable, and to produce simple node-set, text, and data results on the side. The foregoing expression, for instance, may be rendered as follows in XML Query:

```
FOR $book IN document
("books.xml")//book,
$author IN
document("authors.xml")//author
WHERE
$author="Kurt Cagle" AND
$book/isbn = $author/isbn
RETURN
$book/title
```

This still uses XPath (which is considered a part of XML Query), but the syntax is considerably more legible, if somewhat more verbose. Indeed, you can see echoes of SQL here:

```
SELECT title
FROM books.book
WHERE authors.author.isbn=
books.book.isbn AND authors.author=
'Kurt Cagle'
```

The result of the query operation returns a set of XML nodes, though, unlike XPath, the possible set of items returned can be considerably more complex – not only scalar types such as dates, numbers, monetary amounts, or strings, but also more formal complex types that are definable within XML Schema. It is this ability to return more formal types that makes XML Query attractive to database developers, especially given that both XPath and XSLT are essentially type-agnostic languages.

XML Query actually consists of a number of different languages that interrelate with one another. The first is actually an extended version of XPath that can be used to reference any specific node or set of nodes in a document tree. In addition to the base functions of XPath, however, XML Query also uses two new operators, the RANGE operator and the dereference operator.

RANGE is primarily a shortcut. For instance, the expression:

```
document("books.xml")//book[RANGE 2
to 5]/title
```

will retrieve the title of the second through fifth book in the node-set selected by the XPath expression, and is basically the same as:

```
document("books.xml")//book[positi-
tion() >= 2 and position() <=
5]/title
```

The dereference operator (“->”), on the other hand, is a powerful new construct that makes it possible to use a reference to retrieve a specific node in a manner similar to that used by the id() function or the set of id and idref attributes. For instance, suppose you had a catalog that had the structure given in Listing 1. You could then retrieve a pointer from a given author to a list of all possible book titles:

```
//author[.='Kurt Cagle']/isbn/@refid ->
book/ title
```

In essence, such a pointer works in the same way that the id() function does in XPath expressions, but is designed to be a little more legible, especially when more than one dereference is involved. Note that it does, however, require a schema declaring the @refid and @isbn attributes as being of type IDREF and ID, respectively.

Pretty FLWRs

If XML Query were simply an extension of XPath, it would seem to make more sense to make an XPath 2.0 iteration. The primary reason for developing XPath is still legibility, which it does by creating a structure that has a fair amount of resemblance to SQL. Specifically, Query utilizes four new key structures: FOR, LET, WHERE, and RETURN, which together are known as the FLWR (pronounced *flower*) language.

With FLWR you can define variables that contain node-sets, scalar or complex data types that let you do a certain amount of intermediate processing. Additionally, you can perform conditional tests, use the core function set of XPath along with a number of additions, and even define your own functions. Finally, FLWR structures are defined to produce output, and in that respect they perform in a manner comparable to simple XSLT templates.

FLWR gives you two operators for defining variables. The LET operator lets you define either a node-set or a scalar variable, something analogous to an XSLT <xsl:variable> call. For instance, to retrieve

all books in the file books.xml, you may have an expression that looks like:

```
LET $books:= document
("books.xml")//book
```

Note that such an assignment means that the variable \$books now contains a node-set and doesn't perform any explicit iteration. You can, however, iterate through each book in a set of books using the FOR operator:

```
FOR $book IN
document("books.xml")//book
```

In this case the function iterates over each node in the XPath expression and assigns it to the variable (here \$book), which can then be manipulated to produce output:

```
FOR $book IN document
("books.xml")//book
RETURN $book/title
LET and FOR can be combined:
```

```
LET $books:= document
("books.xml")//book
FOR $book IN $books
RETURN $book/title
```

Here, books contains a node-set of book nodes that are then iterated via the FOR \$book IN \$books statement. Note that this is basically analogous to the <xsl:for-each> statement in XSLT.

The WHERE statement in turn performs much the same function as a predicate expression in XPATH or a WHERE statement in SQL. It effectively provides a way to qualify the set, pruning out those nodes that don't satisfy the given condition. The WHERE statement is optional, under the assumption that if it isn't included, all nodes defined in the preceding FOR or LET statements are then passed to the RETURN operator.

The RETURN portion of an XML Query operator handles the actual generation of the resultant code. Typically, an XML Query will itself be contained within the body of an XML document, depending on the parsing processor, and the RETURN body in turn performs a function analogous to an XSL template. The following code, for example, will generate a list of books that contain XML somewhere in the title of the book:

```
<catalog>
FOR $book IN document
("books.xml")//book
WHERE contains($book,'XML')
RETURN $book
</catalog>
```

It's worth mentioning that you can have additional FLWR expressions within the RETURN portion of a given FLWR expression. For example, the code in Listing 2 would order books in descending order, by title, for each unique publisher.

This example, taken from the XML Query working draft, illustrates the use of a number of functions – distinct(), which eliminates duplicate nodes, and the SORTBY operator, which sorts elements based on a specific element or attribute (here, price for \$book and name for

\$publisher). Note that such a query is still a well-formed XML expression.

You can also use conditional expressions in your output, with IF/THEN/ELSE (see Listing 3).

Data Types and Functions

Two of the charges frequently leveled at XPath are its inability to define internal functional notation and its extremely limited type support (in essence, node-sets, nodes, strings, and numbers). This is in fact one of the primary motivators for an XML Query language, though it's likely that XPath 2.0 will also support both types and user functions.

XML Query will lean heavily on the XML Schema specifications (especially Schema[2], Data types) for working with data types, though that support is currently one of the least well defined areas in the specification. Because XML Query performs sorting and comparison operations, it's up to the particular schema to define the ordering mechanism for a given type – whether one expression is “less than” or “greater than” another expression for the specified type.

In addition to making it easier to work with types such as dates, this mechanism will also make it possible to create casts that are analogous to casting from one type to another in a strongly typed language such as C++. At a minimum, this will make it easier to compare an integer and a floating point number, but it will really come into its own when it becomes possible for a given set of XML to inherit a complex data type and then cast back to an abstract type. For instance, both an American and an English address block can be cast back to a generic address for purposes of generating queries.

XML Query will also make it possible to create functions that can be invoked from within XQuery/XPath expressions. As an example, the depth() function could be defined to ascertain the maximum depth (the number of nodes from the root to the farthest leaf) for a given node (see Listing 4).

Notice that this particular function is recursive; it calls itself with the children of the current element until no node is found that has any more children. Such a definition is highly procedural in its form, complete with the return data type given as an integer. There are provisions in the language to handle the output to user-defined types as well, via schemas defined within the XQuery call.

XML Query vs XSLT?

XML Query is a language for performing queries against XML data sources and

generating result node-sets in XML or related languages. XSLT is a language for processing XML data sources and generating result node-sets in XML or related languages. In other words, there's a lot of overlap between the two specifications, a point that has caused no little consternation among the users of XSLT.

In both cases there is a requirement for a processor that actually performs the filtering or transformation based on the XSLT or XQuery code. An interesting exercise would be to use something like a regular expression engine to convert XQuery into augmented XSLT code, but it's likely that XQuery will quickly be adopted by most of the major SQL vendors.

The two operate within similar niches. The primary differences between the two stem from the way programming is done with them. XML Query is a very procedural solution to XML manipulation, although, like XSLT, it operates on sets of nodes rather than on single instances of data. XSLT, on the other hand, is very much oriented around pattern-matching algorithms, making for programs that are comparatively terse yet very powerful.

Perhaps the best way to explain the difference between the two would be to see XML Query as the C++ solution to XML programming, oriented toward solving problems in a very linear fashion, while XSLT is much more akin to working with regular expressions in a language like Perl. One regular expression, especially applied recursively, can typically perform as much work as a thousand-line program written in C++. However, this power comes at the cost of terseness, difficulty in writing, and, occasionally, performance. The same thing holds true of XSLT, which is capable of incredibly sophisticated behavior but often at the cost of being unintelligible to the average user.

I suspect that the relationship of XML Query to XSLT will in fact develop much the same basic synergy as Visual Basic has with C++. For simple transformations (those that typically involve fairly linear mapping) XML Query will probably end up being the preferred mechanism, especially when working against a database. XSLT, however, will likely become the power tool application of the Internet, handling routing, sophisticated transformations of XML sources, and the creation of complex applications. Either way, the upshot will likely be even more development moving into the realm of XML from both the COM and Java sides. ☺

KURT @ KURTCAGLE.NET

LISTING 1

```
<catalog>
<books>
  <book isbn="122536">
    <title>XML Developers Handbook</title>
  </book>
  <book isbn="3155232">
    <title>Advanced XSLT</title>
  </book>
</books>
<authors>
  <author>
    <name>Kurt Cagle</name>
    <books>
      <isbn refid="122536"/>
      <isbn refid="3155232"/>
    </books>
  </author>
</authors>
</catalog>
```

LISTING 2

```
<publishers>
FOR $publisher in distinct(document
("books.xml")//publisher)
RETURN
<publisher>
<name>$publisher/text()</name>,
FOR $book IN document("books.xml")//book
[publisher=$publisher]
RETURN
<book>
  $book/title,
  $book/price
</book> SORTBY(price DESCENDING)
</publisher> SORTBY(name)
</publishers>
```

LISTING 3

```
<publishers>
FOR $publisher in distinct(document
("books.xml")//publisher)
RETURN
<publisher>
<name>$publisher/text()</name>,
FOR $book IN document("books.xml")//book
[publisher=$publisher]
RETURN
<book>
  $book/title,
  $book/price
  IF number($price) > 50
  THEN <category>Expensive</category>
  ELSE <category>Inexpensive</category>
</book> SORTBY(price DESCENDING)
</publisher> SORTBY(name)
</publishers>
```

LISTING 4

```
NAMESPACE xsd="http://www.w3.org/2000/10/
XMLSchema-datatypes"

FUNCTION depth(ELEMENT $e) RETURNS xsd:integer
{
  -- An empty element has depth 1
  -- Otherwise, add 1 to max depth of children
  IF empty($e/*) THEN 1
  ELSE max(depth($e/*))+1
}

depth(document("books.xml"))
```

DOWNLOAD THE CODE @
www.xml-journal.com



Plan to Exhibit

Provide the Resources To Implement Wireless Strategy

The conference will motivate and educate. The expo is where attendees will want to turn ideas into reality. Be ready to offer solutions.

INTERNATIONAL

WIRELESS BUSINESS & TECHNOLOGY

CONFERENCE & EXPO

CONFERENCE & EXPO
CONFERENCE & EXPO



Shaping Wireless Strategy for the Enterprise

Santa Clara, CA

May 7-9, 2002

WirelessEdge will provide the depth and breadth of education and product resources to allow companies to shape and implement their wireless strategy. Developers, i-technology professionals and IT/IS management will eagerly attend.

Plan to Attend the 3-DAY Conference

WHO SHOULD ATTEND

Mobile & Wireless Application Professionals who are driving their enterprise's wireless initiatives:

- Program Developers
- Development Managers
- Project Managers
- Project Leaders
- Network Managers
- Senior IT and Business Executives

Conference Tracks

Track One: Development

WAP
i-Mode
Bluetooth / 802.11
Short Messaging
Interactive Gaming
GPS / Location-Based
Wireless Java
XML & Wireless Technologies

Track Two: Connectivity

Smart Cards
Wireless LANs incl. Bluetooth
UMTS/3G Networks
Satellite Broadband

Track Three: Wireless Apps

Education
Health Care
Entertainment
Transport
Financial Services
Supply Chain Management

Track Four: Hardware

Cell Phones/
WorldPhones
PDAs
Headphones/
Keyboards /
Peripherals
Transmitters/
Base Stations
Tablets

Track Five: Business Futures

Wireless in Vertical Industries
The WWW
Unwired Management
From 3W to 4W: Issues and Trends
"Always-On" Management
Exploiting the Bandwidth Edge
Unplugged Valueware
Wireless Sales & Marketing

FOR INFORMATION CALL

201 802-3069

SPEAKER PROPOSALS INVITED

WWW.SYS-CON.COM

SHAPE YOUR WIRELESS STRATEGY SAVE THE DATES!




EXCLUSIVE SPONSORSHIPS AVAILABLE

Rise above the noise. Establish your company as a market leader. Deliver your message with the marketing support of

SYS-CON MEDIA

SYS-CON EVENTS

THE LARGEST WEST COAST WIRELESS BUSINESS & TECHNOLOGY CONFERENCE OF THE YEAR!



**How it all
fits
together**

The Brave New World of Web Services:

SOAP, WSDL, and UDDI PART 2

In the complex business world of service, organizations need to lower costs and find efficient business delivery models. This means they often outsource parts of the service delivery business. For example, they

may preserve customer relations personnel and the customer relationship, but outsource the actual work. New information needs to be communicated from one organization to another.



Ron Ben-Natan is chief technology officer at ViryaNet Inc., a software provider of wireless workforce-management and field-service solutions. Previously he worked for companies such as AT&T Bell Laboratories and Merrill Lynch, and as a consultant at JP Morgan. Ron has a PhD in computer science in the field of distributed computing and has been architecting and developing distributed applications for almost 20 years. He is the author of CORBA, Objects on the Web, and CORBA on the Web, and coauthored IBM San Francisco Developer's Guide and IBM WebSphere Starter Kit.



written by Ron Ben-Natan

In the “old” world this would involve point-to-point integration between various systems, which would be extremely painful. It would involve a lock-in to one workforce-delivering company and would preclude, for example, bidding out the work, working with multiple providers of workforce services, and, obviously, the notion of workforce marketplaces.

Since these concepts are the crux of B2B interactions and the true enablers of business efficiencies, they are of fundamental interest to the service industry. The examples in this article involve the transmission of “service order” information.

In one scenario an organization captures order-related information, such as customer location, the required service to be performed, and various entitlement terms (such as when the service should be provided). Once the company owning the customer relationship captures this information, it communicates with a workforce provider and creates a call entity. This is done by sending a SOAP request, that is, activating a Web service provided by the second company. The Web service is defined in WSDL (Web Services Description Language) and discovered through the use of a UDDI registry.

In Part 1 of this series we examined the Web services model, SOAP, WSDL, and the UDDI mechanism. In this article we delve into more detail and offer some examples.

The goal is to provide the basics of how the three technologies fit together and how they'd be used, and a general understanding of the structures in each technology set. You won't master these technologies; for that you should read the actual specs and examples available at www.soap.org, www.wSDL.org and www.uddi.org, or on IBM's and Microsoft's Web sites.

In this article the examples are taken from ViryaNet's Service Hub platform – a solution used by companies in the business of service delivery (companies doing installations in homes and businesses, companies that dispatch technicians for fixing and replacing equipment, companies that manage a distributed workforce of claims adjusters or other service personnel).

Businesses in such industries are constantly looking for more efficient business models that often require cooperative business processes between disparate organizations – environments perfect for reaping the benefits of Web services.

SOAP Messaging

Listing 1 shows a SOAP message for creating a service order within the ViryaNet Service Hub. The scenario in which such a document would be used is one in which an external system sends a message to the Service Hub – specifically, to the workforce management system. The external system can be a provisioning system or a trouble-ticket management system in the telecommunications industry, or a help-desk system or call-center system in other verticals.

The information passed into the Service Hub describes the customer, the equipment, and the service requested. This is then sent to the workforce management system so field engineers can be dispatched to the field to work the order.

As can be seen from Listings 1 and 2, SOAP messages are usually encapsulated in HTTP requests and responses. The data itself is an XML document and is encapsulated inside an XML envelope defining the routing and messaging information.

SOAP messages typically have three parts: an envelope, a header, and a body. The envelope and the body are mandatory elements within the SOAP message; the header is optional. The envelope is the root element of the XML document and the body is the payload of the message. The body is a child element in the envelope.

The header element is a generic mechanism for adding attributes and properties to the messaging without requiring (1) prior centralized agreement between communicating parties and (2) some form of federation for messaging patterns. Header attributes are used by recipients to tell them how to process the messages.

A header in a SOAP message may have various attributes. One is the `mustUnderstand` attribute. By tagging a header element with a “1” value for this attribute, the sender language is implying a processing pattern that requires that the semantics of that element be obeyed. If the receiver can't abide by that directive, it must fail in the processing of the message.

Another important attribute for header elements is the actor attribute. SOAP documents can be passed from a sender to an ultimate receiver in a way that passes through multiple parties that may or may not do some processing based on the SOAP document. These can be technical “actors” that perform simple things like routing or may involve message semantics. The actor attribute allows the tagging of header elements in a way that forces an intermediary to process the element and remove it from the SOAP document.

Encoding of data within a SOAP message is fairly straightforward – and similar to most information systems, such as databases and object-oriented languages. Types are either primitive or compound and may be constructed from several parts. This is a recursive structure in which the compound types are made from other compound types or primitive types. Eventually everything is simply one big tree (or hierarchy) where the leaf nodes are simple (primitive) types.

The types themselves follow the XML Schema methodology and allow the messages to be truly self-describing. Although this is an assumption that SOAP is built on, it can't be called an inherent part of SOAP. SOAP is more focused on the messaging and routing elements, and makes use of XML and XML Schema for the body of the message; this is often called *the message payload*. An example of the generic encoding method (without the use of a Schema) follows:

```
<SOAP-ENV:Body>
  <ServiceOrder>
    <ID>213408798</ID>
    <CustomerRef>Fleet<CustomerRef>
    <TimeWindow>

    <From>2001:4:6:13:0:0</From>
    <To>2001:4:6:15:0:0</To>
  </TimeWindow>
  ...
</ServiceOrder>
</SOAP-ENV:Body>
```



SOAP is not directly related to HTTP. It can be delivered on multiple transports using multiple protocols, not just HTTP. Still, HTTP plays a central role in the life of SOAP. Remember, the main theme is that of XML over HTTP. So while SOAP messages can be sent over additional protocols (and mail protocols are certainly being used), the major binding of SOAP to a transport is to HTTP. Therefore, a large part of the SOAP specification document is dedicated to describing the use of SOAP in HTTP.

SOAP makes use of the HTTP headers in many ways, but adds a few that are SOAP specific. A SOAP message in an HTTP request/response pair always uses the text/XML content type, as mandated by the SOAP specification. Other than that, most headers remain the same. One additional header is the `SOAPAction` header field that specifies a URI that hints at the intent of the message. SOAP responses use the same structure as HTTP responses, including the header.

Apart from using general HTTP requests and responses, SOAP also defines an HTTP extension framework. Messages sent in this way force the receiver of the SOAP messages to manage the processing of this request as a SOAP operation as opposed to letting the receiver decide what to do with the request. Basically, the difference is the use of an M-POST request type instead of a POST request type. For example, the message shown in Listing 1 will appear as shown in Listing 3 when invoked within the HTTP extension framework.

WSDL

WSDL is an XML format for describing services on the Web. It views a service as a set of endpoints that communicate by passing XML documents or through an invocation paradigm. WSDL describes services both abstractly and as a concrete implementation. Since WSDL follows the same general pattern as SOAP, the WSDL specification describes how it can be used in a SOAP environment. While WSDL is not limited to use

with SOAP, and is protocol and platform agnostic, people writing software that conforms to WSDL are probably using SOAP.

WSDL describes seven abstractions: services, ports, bindings, port types, operations, messages, and types.

A *service* is the entity that is the focus of what WSDL is all about, and the purpose of a WSDL document is to describe the structure of a service. A service is described in terms of a collection of ports that represent network endpoints, that is, routines that can communicate over the Web to request and provide the service.

A *port* is defined in terms of a combination of an address on the Web along with a binding as the concrete protocol and data format that is supported by the endpoint. The *binding* is described by a more abstract entity – the *port type* that itself is described as a collection of *operations*. The data exchanged by the endpoints is also described in an abstract manner by *messages*.

Finally, the definition of the data being passed within a message is encapsulated within *type* definitions. *Types* in WSDL are described according to the XML Schema specification. WSDL defines a set of specific bindings in addition to the general binding method. The extensions of concrete bindings are for SOAP, HTTP GET and POST requests, and MIME.

Listing 4 shows the canonical service document structure as defined by the WSDL specification. Listing 5 is an example WSDL document describing the order creation service in ViryaNet's Service Hub platform. Note that services are defined using the following major element categories:

- Types that define the data types used in the messages
- Messages that logically define what is transmitted between the endpoints
- Port types that specify the abstraction of the operation
- Bindings that define the concrete protocol and data format
- Ports that specify the address for the binding
- And finally, the service definition itself that aggregates all of the above

Types in WSDL are XML Schema types. Messages consist of one or more logical parts, each one having a type. Message parts have a name and an element; the element attribute specifies a type that is defined in the document. Messages, therefore, define the abstractions in terms of the data types used in communication between the two systems. A port type is a named set of abstract operations and messages. Each port type has a name and some optional attributes, such as an attribute that defines if the endpoint can support one-way, request-response, solicit-response, and notification-type communications.

If the port type defines a one-way operation, the definition will take the form:

```
<wsdl:portType>
  <wsdl:operation name="createCall">
    <wsdl:input .../>
  </wsdl:operation>
</wsdl:portType>
```

If the port type defines a request-response or a solicit-response operation, the definition will take the form:

```
<wsdl:portType>
  <wsdl:operation name="createCall">
    <wsdl:input ... />
    <wsdl:output ... />
    <wsdl:fault ... />
  </wsdl:operation>
</wsdl:portType>
```

If the port type defines a notification operation, the definition will take the form:

```
<wsdl:portType>
  <wsdl:operation name="createCall">
    <wsdl:output .../>
  </wsdl:operation>
</wsdl:portType>
```

Bindings describe the message formats and protocol details for operations and messages defined by a particular port type. There may be numerous bindings for a single port type. Ports are the physical endpoint descriptors and specify the physical location from which the service may be received. Finally, services group a set of related ports together and effectively define a domain where a service is provided.

Since WSDL is in many ways so close to SOAP, the WSDL specification has a special section describing a SOAP binding. The SOAP-specific definitions include a way to specify that a binding is bound to SOAP, a way to specify an address for a SOAP endpoint, a way to use a SOAPAction URL in an operation definition, a set of definitions for headers transmitted as part of a SOAP envelope, and a way for specifying SOAP roots in XSD.

Each of these specifications makes use of a SOAP namespace. For example, a SOAP binding element can be used to specify that SOAP is being used as the underlying protocol. In this case the WSDL binding section will take the form:

```
<binding ... >
  <soap:binding ... />
</binding>
```

In the same way, when an operation is defined using the SOAP extension, the operation takes the form:

```
<operation ... >
  <soap:operation soapAction="..." ... />
</operation>
```

Other similar examples exist, including soap:body, soap:fault, soap:header, and soap:address.

Another part of the WSL specification defines an HTTP GET/POST binding that allows for a more primitive implementation. Here, HTTP is used for the messaging transport and the service definer can specify a binding using HTTP GET or HTTP POST, can specify that the port is implemented as an HTTP endpoint, and can specify an address for each operation relative to the HTTP endpoint.

UDDI

UDDI defines a way to publish and discover information about Web services. It defines infrastructure that's mandatory for the development of e-commerce, collaborative marketplaces, and the like. It's a tool for developers and designers more than anything else, and helps standardize how services can be defined and discovered. Thus it complements WSDL and SOAP in the sense that it's fairly logical to see the definitions themselves defined in WSDL, activated using SOAP, but registered and discovered using UDDI.

The focus of UDDI is the registration and discovery of services. UDDI relies on the existence of a distributed registry that is implemented in XML and communicated with using XML. UDDI business registration is done using an XML file that describes a business entity and the associated Web services.

There are three parts to such definitions – three parts that are equivalent to the real-world services provided by “white pages,” allowing address and contact information to be discovered; “yellow pages,” allowing categorizations; and “green pages,” exposing technical information about services that are being exposed.

All of this information is maintained within the UDDI registry on the Web. UDDI defines the XML standards allowing developers to register information within the registry and to discover and use information stored within the registry.

In a typical scenario using UDDI, one party registers information about the Web services supported in a system. The information is added to the UDDI registry through a Web site or by using tools that make use of the programmatic APIs defined by the UDDI specification. The UDDI registry is logically one database, but physically may be (and most probably is) distributed through sets of physical registries (after all, it has to scale well). UDDI doesn't focus on the discovery stage per se, and doesn't mean to replace search engines and portals. It merely defines the structure through which such programs can look up information.

Technically, UDDI consists of an XML Schema (XSD) for SOAP messages and a definition of APIs for performing the UDDI operations. The schema definitions allow a programmer to define business information, service information, binding information, and information about specifications for services.

Business information takes the form of the businessEntity elements that support “yellow pages” taxonomies so that searches can be performed. Substructures of the businessEntity element also define the information required to support “green pages” type functions.

Service information is described by the businessService element. Such elements are higher level elements. Within each one of the businessService elements, many Web service descriptors can exist. Such an element allows segmentation and categorization at a higher level. The bindingTemplate element allows programmers to provide information about the addresses through which a Web service can be contacted.

A businessEntity structure typically represents information about a business as well as the services it offers. This includes the business name, a unique identifier for the business, a description of the business entity, contact information, and, in particular, the list of business-Services supported.

A businessService has a name and a description, a unique key, and, most important, the list of bindingTemplates. The bindingTemplate also has a set of descriptors as well as a required element called accessPoint that describes the endpoint access. This element also points at the tModel entity that defines the actual technical fingerprint of the service.

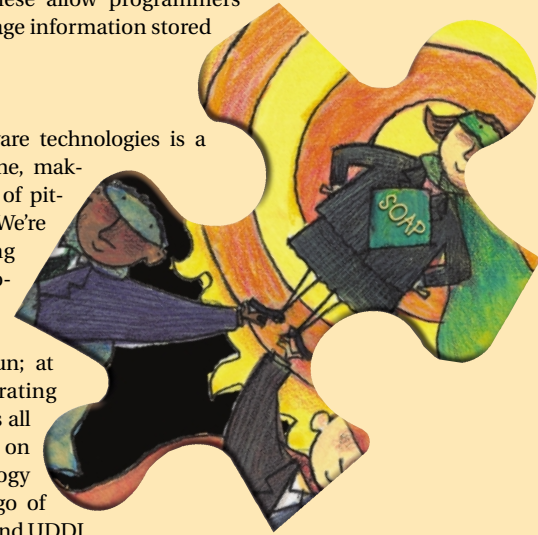
In terms of the UDDI API itself, it's a SOAP-based API, meaning that every invocation takes the form of a SOAP message. The requests typically define which function is requested. This is passed to a UDDI registry provider that replies with a SOAP document. The API consists of more than 30 SOAP messages that can be partitioned into three groups:

1. **Browse APIs:** These APIs find elements and the information associated with entries in a UDDI registry. This category consists of find_xx APIs.
2. **Drill-down APIs:** The elements in a UDDI registry are organized in hierarchies. Once we can find elements using the find_xx APIs, we can further navigate the hierarchy using get_xx APIs.
3. **Publishing APIs:** These allow programmers and systems to manage information stored in a UDDI registry.

Final Word

The world of software technologies is a constantly changing one, making it exciting and full of pitfalls at the same time. We're constantly discovering that emerging technologies mean we have to constantly update our skills. Sometimes it's fun; at other times it's frustrating and difficult. In fact, it's all too easy to just give up on one of these technology waves. Well, don't let go of this one: SOAP, WSDL, and UDDI hold too much promise and are quickly becoming the cornerstone of too many jobs. Take my advice: learn them and implement them. ☘

RON.BEN-NATAN @ VIRYANET.COM



LISTING 1 Example SOAP message – request

```
POST /CreateCall HTTP/1.1
Host: virya.viryanet.com
Content-Type: text/xml; charset="utf-8"
Content-Length: 953
SOAPAction: http://virya.viryanet.com/
servlet/ServletBroker?
provider=Call&service=createCall

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/
    envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/
    encoding/">
  <SOAP-ENV:Body>
    <viryanet:CreateCall xmlns:
    viryanet=
      "http://virya.viryanet.com/servlet/
      ApiRepository">
      <problemDescription type="STRING">
        <![CDATA[BURNING SMELL]]>
      </problemDescription>
      <warrantyFlag type="BOOLEAN">
        false
      </warrantyFlag>
      <subContractorFlag type="BOOLEAN">
        false
      </subContractorFlag>
      <equipmentNumber type="STRING">
        <![CDATA[1234]]>
      </equipmentNumber>
      ...
      <projectId type="STRING">
        <![CDATA[2134]]>
      </projectId>
      <actualResponseTime type="STRING">
        <![CDATA[17 Nov 2000 7:00]]>
      </actualResponseTime>
      <downFlag type=
        "BOOLEAN">false</downFlag>
      <viryanet:CreateCall>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

LISTING 2 Example SOAP message – response (including input data)

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: 423

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/
    envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/
    encoding/">
  <SOAP-ENV:Body>
    <viryanet:CreateCall xmlns:
    viryanet=
      "http://virya.viryanet.com/servlet/
      ApiRepository">
      <problemDescription type="STRING">
        <![CDATA[BURNING SMELL]]>
      </problemDescription>
      ...
      <projectId type="STRING">
        <![CDATA[2134]]>
      </projectId>
      <downFlag type="BOOLEAN">
        false
      </downFlag>
      <viryanet:CreateCall>
      <viryanet:CreateCallResult>
        <date null="" type="NULL"/>
        <status type="STRING">
          <![CDATA[0]]>
        </status>
        <id type="STRING">
          <![CDATA[886]]>
        </id>
      </viryanet:CreateCallResult>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

LISTING 3 M-post SOAP request

```
M-POST /CreateCall HTTP/1.1
Man: http://schemas.xmlsoap.org/soap/
envelope/
```

```
/ns=VN
Content-Type: text/xml; charset="utf-8"
Content-Length: 953
VN-SOAPAction:
http://virya.viryanet.com/servlet/
ServletBroker?
provider=Call&service=createCall

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/
    envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/
    encoding/">
  <SOAP-ENV:Body>
    <viryanet:CreateCall xmlns:viryanet=
      "http://virya.viryanet.com/servlet/
      ApiRepository">
      <problemDescription
        type="STRING">
        <![CDATA[BURNING SMELL]]>
      </problemDescription>
      ...
      <downFlag type="BOOLEAN">
        false
      </downFlag>
      <viryanet:CreateCall>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

LISTING 4 Service document structure in WSDL

```
<wsdl:definitions
  name="nmtoken"?
  targetNamespace="uri"?>

  <import namespace="uri"
    location="uri"/>*

  <wsdl:documentation/?>

  <wsdl:types?>
    <wsdl:documentation/?>
    <xsd:schema/?>*
    <-- extensibility element -->*
  </wsdl:types>
```



```

<wsdl:message name="nmtoken">*
  <wsdl:documentation/?>
  <part
name="nmtoken"
element="qname"?>*
  </wsdl:message>

  <wsdl:portType name="nmtoken">*
    <wsdl:documentation/?>
    <wsdl:operation name="nmtoken">*
      <wsdl:documentation/?>
      <wsdl:input
name="nmtoken"?
message="qname"?>
        <wsdl:documentation/?>
        </wsdl:input>
        <wsdl:output
name="nmtoken"?
message="qname"?>
          <wsdl:documentation/?>
          </wsdl:output>
          <wsdl:fault
name="nmtoken"
message="qname">*
            <wsdl:documentation/?>
            </wsdl:fault>
            </wsdl:operation>
          </wsdl:portType>

  <wsdl:binding name="nmtoken"
type="qname">*
    <wsdl:documentation/?>
    <!-- extensibility element -->*
    <wsdl:operation name="nmtoken">*
      <wsdl:documentation/?>
      <!-- extensibility element -->*
      <wsdl:input name="nmtoken"?>
        <wsdl:documentation/?>
        <!-- extensibility element -->
      </wsdl:input>
      <wsdl:output name="nmtoken"?>
        <wsdl:documentation/?>
        <!-- extensibility element -->
      </wsdl:output>
      <wsdl:fault name="nmtoken">*
        <wsdl:documentation/?>
        <!-- extensibility element -->
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="nmtoken">*
    <wsdl:documentation/?>
    <wsdl:port
name="nmtoken"
binding="qname">*
      <wsdl:documentation/?>
      <!-- extensibility element -->
    </wsdl:port>
    <!-- extensibility element -->
  </wsdl:service>

  <!-- extensibility element -->*
</wsdl:definitions>

```

LISTING 5 WSDL definition for a createCall message (creating a service order)

```

<?xml version="1.0"?>
<definitions name="Call"

targetNamespace=
"http://virya.viryanet.com/servlet/
ApiRepository/call.wsdl"

xmlns:tns=
"http://virya.viryanet.com/servlet/
ApiRepository/call.xsd"

xmlns:xsd_vn=
"http://virya.viryanet.com/servlet/
ApiRepository/call.wsdl"

```

```

xmlns:soap=
"http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/"

  <types>
    <schema targetNamespace=
"http://virya.viryanet.com/servlet/
ApiRepository/call.wsdl"
xmlns="http://www.w3.org/1999/XMLSchema">
      <element name="CallRequest">
        <complexType>
          <all>
            <element
name="problemDescription"
type="string"/>
            <element
name="warrantyFlag"
type="boolean"/>
            <element
name="subContractorFlag"
type="boolean"/>
            ...
            <element
name="projectId"
type="string"/>
            <element
name="downFlag"
type="boolean"/>
          </all>
        </complexType>
      </element>
      <element name="CallResponse">
        <complexType>
          <all>
            <element
name="date"
type="string"/>
            <element
name="status"
type="string"/>
            <element
name="id"
type="string"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>

  <message name="InputCall">
    <part
name="body"
element="xsd_vn:CallRequest"/>
  </message>

  <message name="OutputCall">
    <part
name="body"
element="xsd_vn:Response"/>
  </message>

  <portType name="CreateCallPortType">
    <operation name="CreateCall">
      <input message="InputCall"/>
      <output message="OutputCall"/>
    </operation>
  </portType>

  <portType name="UpdateCallPortType">
    <operation name="CreateCall">
      <input message="InputCall"/>
      <output message="OutputCall"/>
    </operation>
  </portType>

  <binding
name="CreateCallSoapBinding"
type="CreateCallType">
    <soap:binding
style="document"
transport=
"http://schemas.xmlsoap.org/soap/http"/>
    <operation name="CreateCall">

```

```

      <soap:operation soapAction=
"http://virya.viryanet.com/servlet/
ServletBroker?
provider=Call&service=createCall"/>
      <input>
        <soap:body
use="literal"
namespace=
"http://virya.viryanet.com/servlet/
ApiRepository/call.xsd"
encodingStyle=
"http://schemas.
xmlsoap.org/
soap/encoding/" />
      </input>
      <output>
        <soap:body
use="literal"
namespace=
"http://virya.viryanet.com/servlet/
ApiRepository/call.xsd"
encodingStyle=
"http://schemas.xmlsoap.org
/soap/encoding/" />
      </output>
    </operation>
  </binding>

  <binding
name="UpdateCallSoapBinding"
type="UpdateCallType">
    <soap:binding
style="document"
transport=
"http://schemas.xmlsoap.org/soap/http"/>
    <operation
name="CreateCall">
      <soap:operation soapAction=
"http://virya.viryanet.com/servlet/
ServletBroker?
provider=Call&service=updateCall"/>
      <input>
        <soap:body
use="literal"
namespace="http://virya.viryanet.com/
servlet/
ApiRepository/call.xsd"
encodingStyle=
"http://schemas.xmlsoap.org/
soap/encoding/" />
      </input>
      <output>
        <soap:body
use="literal"
namespace=
"http://virya.viryanet.com/servlet/
ApiRepository/call.xsd"
encodingStyle=
"http://schemas.xmlsoap.org/
soap/encoding/" />
      </output>
    </operation>
  </binding>

  <service name="CallService">
    <documentation>
Call creation/update service
    </documentation>
    <port
name="CreateCallPort"
bindings="tns:CreateCallSoapBinding ">
      <soap:address location=
"http://virya.viryanet.com/
servlet/ServletBroker"/>
    </port>
    <port
name="UpdateCallPort"
binding="tns:UpdateCallSoapBinding ">
      <soap:address location=
"http://virya.viryanet.com/
servlet/ServletBroker"/>
    </port>
  </service>
</definitions>

```

Epic Editor 4.2

by Arbortext



AUTHOR BIO

Kristian Cibulskis is the CTO of Vertica Systems, which provides XML- and Java-based data integration solutions for clinical trials. He holds a BS in computer science from Cornell University.

KCIBULSKIS @ VERTICASYSTEMS.COM

—[REVIEWED BY KRISTIAN CIBULSKIS]—



SPECS

Arbortext, Inc.

1000 Victors Way

Ann Arbor, MI 48108

Phone: 734 997-0200

Web: www.arbortext.com

E-mail: info@arbortext.com

Test Environment

OS: Windows 2000 running on Dell Latitude

Processor: 450MHz Pentium III

Memory: 256MB RAM

Epic Editor is a full-featured editor for structured content such as SGML/XML that allows authors to focus on content development instead of formatting issues. Using Epic Editor, a team of authors can collaborate on the content of a single master document, which can then be personalized and tailored for delivery through multiple media channels, such as the Web, CD-ROM, print, and wireless devices.

Epic Editor is the core product in the Epic product suite. Other components, such as Epic Print Composer and Epic Web/Wireless Composer, aid in the delivery of content created with Epic Editor. Components for interaction between Epic Editor and content management systems, such as Documentum and Oracle iFS, are also available.

Installation

The installation of Epic Editor was straightforward. In addition to specifying which products I wanted to install, I was able to create a separate working directory for my XML documents. At first the install size – 130MB – of the core product seemed a bit heavy. However, when I dug into the functionality of Epic Editor, I was surprised at how much they had packed into the feature set.

Once the installation process finishes, the Epic License Management software is launched. This is used to unlock the various products you've purchased. Manually entering product keys was a little cumbersome as each key contains 37 digits and there are potentially eight products to license. To be fair, however, it's a onetime process.

Features and Impressions

To get an idea of Epic Editor in a nutshell, think of Microsoft Word built for XML! A plethora of tools are available via the menus and the toolbar, but the first feature that jumps out is a split-screen view including the Document Map and a rendered view of the document. The two views are synchronized so highlighting text, making changes, or scrolling in one window is immediately reflected in the other window.

Editing

The Document Map is a tree-style graphical view of your SGML/XML document that clearly depicts the structure of your document. The editor will only allow you to create well-formed XML, inserting necessary elements as you go.

The most powerful features are available when you have a DTD defined for the document as the entire document is parsed and validated in real time. When you reach the end of an element and hit ENTER, the Quick Tag feature displays a popup menu containing only valid elements according to the DTD. This also applies to cut-and-paste or drag-and-drop operations. After selecting an element, you can drag it to another location in the document. Epic Editor

visually indicates if it's possible to insert the element at the given location according to the rules defined in the DTD. Sometimes an insertion is possible only with the inclusion of another tag, which Epic Editor indicates and then does the work for you. This makes it a breeze to create documents that are compliant with either an industry standard or custom DTD.

While editing your structured content via the Document Map is easy to do, it's even easier to edit in the rendered view. The rendered view window displays your content after applying a style-sheet, giving you a real-time preview of your final output. Making changes in this view is as simple as using a word processor and all changes are immediately visible within the Document Map.

Collaborating

While Epic Editor works well as a stand-alone editing tool, it excels at facilitating collaboration. Content can be searched, checked in, and checked out from a repository without ever leaving Epic Editor. Additionally, for some repositories Epic Editor can automatically break apart structured documents into their elements for storage.

Abortext has chosen to support a best-of-breed repository approach. Currently, the product includes an adapter for connecting to Documentum 4i. An adapter for integration with Oracle iFS is also available as an add-on.


Epic Editor 4.2 now includes a Change Tracking feature, which allows an author to view all the individual modifications and revisions that have been made to a document. This new feature facilitates information management processes such as content creation, review, and approval.

Extending

The Epic Editor exposes a robust API consisting of over 500 functions and 75 events for developers to extend and customize the product to their needs. Supported programming languages include C, C++, Java, JavaScript, Visual Basic, TCL, Perl, and Python as well as a COM interface. Additionally, Abortext has included its custom scripting language, Abortext Command Language (ACL). With ACL you're also able to execute individual commands from within Epic Editor using a simple but effective command-line window.

Conclusion

The real-time parsing and validation of your document against a DTD make Epic Editor a must for creating complex structured documents. By allowing the author to focus on content rather than SGML/XML technical details or formatting, it provides an extremely productive editing environment.

While Epic Editor would be a welcome addition to any individual author's toolbox, it really shines when used to collaborate across multiple authors. With its advanced repository support and focus on content production, Epic Editor makes it easy for large teams to produce structured documentation. 

Premiering...this *fall* subscribe **Now!**

**FOR FAST
DELIVERY**

Go
Online
and
Subscribe
Today!

The World's Leading Independent WebLogic Developer Resource



Helping
you enable
inter-company
collaboration
on a global scale

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

WebLogicDevelopersJournal.com

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

**SYS-CON
MEDIA**

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

**SYS-CON
MEDIA**



WebSphereDevelopersJournal.com

**WebSphere
DEVELOPER'S JOURNAL**



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

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

**Then
Subscribe
Online and
Save \$31!
It's that easy**

The
World's
Leading
Independent
WebSphere
Developer
Resource



What makes service discovery tick?

Deeper into UDDI

Moving from the introduction to the UDDI information model and APIs in last month's *XML-J* (Vol. 2, issue 10), this month we'll take a detailed look at the UDDI categorization and classification mechanisms and the magic of UDDI tModels.

One of the main objectives of the UDDI registries is the discovery of services, whether statically (design time) or dynamically (runtime). This entails a form of search through a large space of entries. In a reasonably populated registry, depending on the search mechanism, this could return a very large set of hits. An early requirement for UDDI therefore was a way to perform intelligent searches. Until we get closer to a truly Semantic Web, the best mechanism to facilitate such searches will continue to be through property-based lookup and taxonomic categorization and classification.

Property-Based Lookup

The simplest type of search involves looking through sets of value spaces. For example, you may want to look for "470-25-4383" in the space of Social Security Numbers (SSNs). Or you may want to look for 02140 in the space of U.S. zip codes. In both cases the result of the search will be a set of entities (a person with the given SSN or all addresses in the 02140 zip code area).

Formalizing the notion of property-based search is relatively easy to do. We need a mechanism to attach any number of properties to a piece of data. Properties need to have names and values. So far so good.

The only complication has to do with supporting a naming system that allows for precision and extensibility without introducing much complexity. Precision of naming has to do with the ability to identify the precise meaning of a property name. Take, for example, a property whose name is *name*. Is this a company's name, a person's name, or the name of a country? XML has addressed this problem for element names with the introduction of namespaces. Namespaces aren't unique to XML. They're extensively used in other areas such as programming languages to pro-

vide specific meaning to simple names. Namespaces allow us to distinguish between the property "name" in the company namespace and the property "name" in the personal information namespace.

In UDDI properties are called *identifiers*. Most UDDI data structures can support any number of properties inside an identifierBag element. The properties are represented by keyedReference elements that have three attributes:

- **keyName (required):** Name of property
- **keyValue (required):** Value of property; can be any string
- **tModelKey (optional):** Universally unique identifier (UUID) of property name namespace that name of property belongs to; the tModel concept will be detailed later

Following is an example of a property identifying a company's Dun & Bradstreet D-U-N-S number:

```
<identifierBag>
  <keyedReference keyName="DUNS" keyValue=
    "00-111-1111" tModelKey="UUID:8609C81E-
    EE1F-4D5A-B202-3EB13AD01823"/>
</identifierBag>
```

In this example the key UUID:8609-C81E-EE1F-4D5A-B202-3EB13AD01823 identifies the space of the D&B D-U-N-S numbers.

Simple property-based searching is easy to do, but it lacks the power to express search constraints that have to do with subselection and membership to multiple categories. For example, if a business manufactures skateboards, it is a manufacturer in the general sense of the term and a manufacturer of light equipment and of sporting goods and of skateboards. To navigate these types of concepts we need category-based search.

Categorization and Classification

Categorization is the process of creating categories, while *classification* is the process of assigning objects to these predefined categories. There are many types of classification schemes. For large information spaces such as businesses and the services they expose, the most useful classification models are hierarchical in nature. UDDI includes in its core specifications three predefined classification schemes:

1. North American Industry Classification System (NAICS) for classifying businesses by industry
2. Universal Standard Products and Services Classification (UNSPSC) for product and service classifications
3. ISO 3166 standard for geographic location classifications

In order to support category-based search, we need to classify items in the search space. We can do this by associating data with the "nodes in various categories" it conceptually belongs to. In UDDI this is done through the categoryBag element. The structure of categoryBag is identical to that of identifierBag: a set of keyedReference elements. In this case, however, the attributes of keyedReference are used in a different way:

- **keyName:** Name of category
- **keyValue:** Value of category according to categorization hierarchy
- **tModelKey:** UUID identifying categorization scheme

This article is based on the UDDI chapter in *Building Web Services* (Sams), a book I've written with Doug Davis, Steve Graham, Yuichi Nakamura, and Ryo Neyama from IBM, Toufic Boubez from Saffron Technology, and Glen Daniels from Macromedia. It's scheduled for release early next month.

AUTHOR BIO

Simeon Simeonov, chief architect at Macromedia, Inc., is a member of the W3C working group on XML protocol and the J2EE expert groups on XML business messaging and XML data binding.

Following is an example of a categorization identifying a company as a sporting goods manufacturer in the New York area:

```
<categoryBag>
  <keyedReference
    keyName="Sporting and Athletic
    Goods Manufacturing" keyValue="33992"
    tModelKey="UUID:C0B9FE13-
    179F-413D-8A5B-5004DB8E5BB2"/>
  <keyedReference
    keyName="New York"
    keyValue="US-NY"
    tModelKey="UUID:4E49A8D6-
    D5A2-4FC2-93A0-0411D8D19E88"/>
</categoryBag>
```

In this example the key UUID:-C0B9FE13-179F-413D-8A5B-5004DB8E-5BB2 identifies the NAICS categories while the key UUID:4E49A8D6-D5A2-4FC2-93A0-0411D8D19E88 identifies ISO 3166 standard for geographic location classifications.

The UDDI tModel Concept

In the UDDI information model described in last month's *XML in Transit* (see Figure 1), business entities expose a number of business services, each of which is accessed via a number of bindings to different protocols and at different physical locations (network addresses). All three information structures – entities, services, and bindings – refer to UDDI technical models (tModels). The tModel structure is a fundamental concept that is at the root of much of the power and flexibility that UDDI provides. As you've seen, tModels are the mechanism used to identify property namespaces and categorization schemes in UDDI.

What Is a tModel?

In order to invoke a service, it's sometimes necessary to know a large amount of details in terms of document formats, transport protocols, business processes, and the like. These details in general could be supplied in the description of the service. If software engineering has taught us one thing in the last decade, however, it is the power of abstraction and reuse. These service specifications could be applicable in several different places for different services, or for different bindings for the same service. They can also be specified by an industry consortium, a standards body, or a large corporation for use by their suppliers. Thus conformance to a known and predefined set of specifications becomes very important. The tModel concept is the mechanism for such abstractions. It allows various enti-

ties (such as businesses, standards bodies, industry groups) to publish abstract specifications to be used by other entities in implementing services.

The tModel Data Structure

The tModel structure is simple and flexible; it allows you to define just about anything. Here are some of the key pieces of information that define a tModel:

- **tModelKey:** Unique identifier for tModel; used in keyedReferences
- **name:** tModel's name
- **description:** Textual description of tModel
- **overviewDoc:** Optional URL to document describing tModel
- **identifierBag:** Optional properties attached to tModel; can be valuable in searching UDDI registry for particular tModel
- **categoryBag:** Optional classification information about tModel; allows tModels to be organized within taxonomies

Using tModels

The simplicity of the tModel data structure makes it a powerful concept, but it can also be a double-edged sword. On the one hand, because of the simplicity in defining a tModel, people can specify anything they want, even if it's not useful to or reusable by anyone else. On the other hand, if used properly, tModels can be used to specify the standards and templates that will be used for the next generation of e-business on the Web.

Throughout the UDDI data structures, tModels are used as references. By convention, the UDDI specification suggests two main uses for them:

1. They are useful in defining namespaces for identifiers and classification, as you've seen in the earlier examples.
2. They're also used in defining what the UDDI documentation calls *technical fingerprints*. This refers to any technical specifications or prearranged agreements on how to conduct business. For example, tModels can be created by

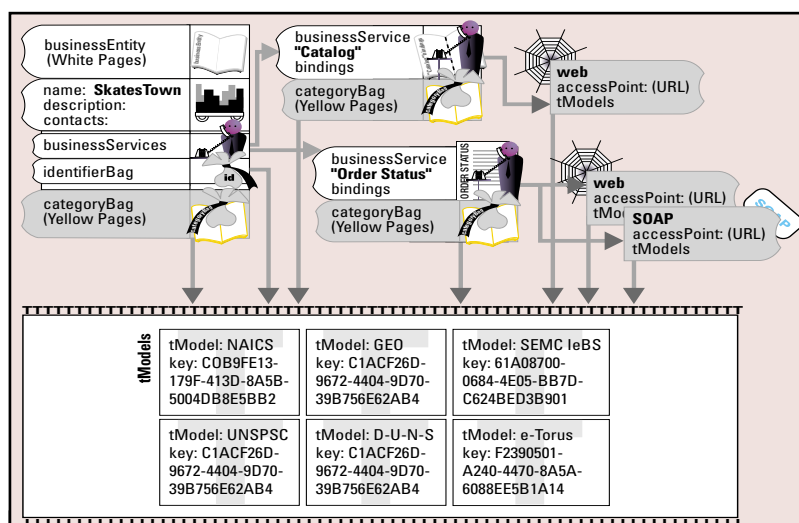


FIGURE 1 UDDI information model

tMODEL: ROOT tMODEL FOR TYPES	
identifier:	unique identifier for UDDI data structure
namespace:	namespace for properties
categorization:	categorization (taxonomy) space
specification:	specification related to Web services
xmlSpec:	specification for Web service using XML messages
soapSpec:	specification for interaction with Web service using SOAP messages
wsdlSpec:	specification for Web service described in WSDL
protocol:	any type of protocol
transport:	wire-level protocol (HTTP, SMTP, etc.)
signatureComponent:	signature component

TABLE 1 Canonical model hierarchy

industry groups such as RosettaNet or by companies wishing to standardize the way they deal with suppliers. As an example, the WSDL (Web Services Definition Language) Schema for Web services descriptions is already defined in UDDI as a tModel. This enables WSDL to be used with UDDI.

Through the containment model of UDDI, tModels can be organized into a hierarchy. UDDI has a number of predefined tModels describing tModel types, called *canonical models*. Each canonical model is a technical fingerprint, a set of well-defined and agreed-upon concepts and rules. The canonical model hierarchy is shown in Table 1, using the tModel's names.

An example of how these models can be put to use are the definitions of the UDDI APIs, which are referred to as tModels because they're well defined and agreed upon through the UDDI specification. These tModels are all classified in the UDDI Type tModel namespace as being of types "specification," "xmlSpec," and "soapSpec." Listing 1 shows the definition of the UDDI inquiry API.

Through the UDDI publishing API, organizations or businesses can define,

create, and publish their own tModels. In fact, they can even create their own taxonomies using tModels.

Third-Party Taxonomies

Taxonomies are the primary means in UDDI of organizing and therefore searching for entries. UDDI version 2.0 added a feature by which third parties could specify new taxonomies and provide a service by which UDDI registries—in particular, operator nodes within the UDDI Business Registry—could validate categorizations asserted against a taxonomy developed by a third party.

Why would anyone want to define a new taxonomy? UDDI provides, by default, the three canonical taxonomies: NAICS, for industry classifications, UN/SPSC, for product/service, and ISO 3166, for geographic classifications. These are general-purpose taxonomies that are quite useful for broadly classifying businesses or services. However, a particular industry might want to define a new taxonomy to detail product classifications according to a scheme commonly used in their industry. A third-party quality of service (QoS) guarantor may wish to define an identifier scheme, classifying businesses and their services according to some new QoS metric for Web services. Organizations may choose to convert their product codes into a taxonomy and categorize which businessService entries are associated with requesting, purchasing, and getting post-sales support on each of their products.

Example Taxonomy

For example, WeMakelt Inc. decides to create a parts code scheme for all of the parts WeMakelt Inc. uses in its manufacturing lines. Because WeMakelt Inc. is a manufacturer of a wide and frequently changing variety of goods, this list of product codes is constantly updated. By creating this taxonomy, the company can make its private UDDI partner catalog registry extremely useful. WeMakelt Inc. will require their suppliers to use Web services to interact with their automated supply re-ordering system. All of the partners will be required to categorize their UDDI businessService entries using WeMakelt Inc.'s custom parts code taxonomy.

In this way the businessService entries in WeMakelt Inc.'s partner catalog UDDI are all classified according to the parts code. When WeMakelt Inc. needs to re-order a part, it can issue a taxonomy-based find operation and retrieve all the business partners that deal in that part. WeMakelt Inc. can then issue RFQs, purchase orders, and the like to resupply a manufacturing line, or to just-in-time prepare for an upcoming manufacturing run.

Let's examine the steps WeMakelt Inc. would use to set up their third-party taxonomy based on the existing product code scheme. The company would create a tModel to describe their product code taxonomy and register it in the UDDI Business Registry using the save_tModel API (see Listing 2).

The UDDI registry will assign a tModelKey, say, UUID:E0AC1230-4CC1-11D5-B353-B4D70FD31643. WeMakelt Inc. would tell all of its suppliers to get this tModel and use it to classify their businessEntity and businessServices according to which of these product codes the supplier provides.

The following is an example of how the parts-ordering service of a midsized widget manufacturer would classify itself within the WeMakelt parts code scheme.

```
<categoryBag>
  <keyedReference
    keyName="Mid-Size Widgets"
    keyValue="123-17-94"
    tModelKey="UUID:E0AC1230-4CC1-11D5-B353-B4D70FD31643" />
</categoryBag>
```

Validating Taxonomies

Because WeMakelt Inc. wants their suppliers properly categorized, it would provide a validation mechanism for its taxonomy. WeMakelt Inc. would set up a Web service to validate any UDDI entry that is categorized using its custom taxonomy. This Web service would conform to the validate_values requirements specified in the UDDI version 2.0 programmer's API. WeMakelt Inc. would register this service with the UDDI Business Registry using the mechanism specified by the operator node that controls WeMakelt Inc.'s entries. It would tell all of its suppliers to reference this validate_values Web service with their private portal UDDI registries.

Without this effort WeMakelt Inc.'s custom taxonomy would have remained unchecked. The difference between a checked and an unchecked taxonomy is fairly important. With a checked taxonomy, any save_business, save_service, or save_tModel that references this taxonomy will use the validate_values service published by WeMakelt Inc. If an entry specifies an invalid category, or WeMakelt Inc.'s validate_values service returns an error for some reason, the save operation for the entry fails. If WeMakelt Inc. develops an unchecked taxonomy, there would be no way its suppliers could ensure that their categorizations accurately reflected WeMakelt Inc.'s supplies coding scheme. ❧

LISTING 1 UDDI inquiry API tModel

```
<tModel tModelKey='uuid:4CD7E4BC-648B-426D-9936-443EAC8AE23'>
  <name>uddi-org:inquiry</name>
  <description xml:lang='en'>UDDI Inquiry API
  - Core Specification
  </description>
  <categoryBag>
    <keyedReference keyName='categorization'
    keyValue='specification'
    tModelKey='uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4' />
    <keyedReference keyName='categorization'
    keyValue='xmlSpec' tModelKey='uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4' />
    <keyedReference keyName='categorization'
    keyValue='soapSpec' tModelKey='uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4' />
  </categoryBag>
</tModel>
```

LISTING 2 UDDI inquiry API tModel

```
<save_tModel generic="2.0" xmlns="urn:uddi-org:api_v2" >
  <authInfo>...</authInfo>
  <tModel tModelKey="">
    <name>WeMakelt Product Code
    Taxonomy</name>
    <description xml:lang="en">...</description>
    <categoryBag>
      <keyedReference
        keyName="TModel categorization
        type"
        keyValue="categorization"
        tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
        <!-- UDDI Types tModelKey -->
      />
    </categoryBag>
  </tModel>
</save_tModel>
```

DOWNLOAD THE CODE @
www.xml-journal.com

SIMEONS @ MACROMEDIA.COM

Dot.com

- buyer's guide
- xml forums
- mailing list
- xml jobs
- xml store

Magazine

- advertise
- authors
- customer service
- editorial board
- subscribe

Content

- archives
- digital edition
- editorial
- features
- interviews
- product reviews
- source code

Conferences

- xml edge
- santa clara, ca
- oct 22-25

Search XML-J

Search

Check out these companies and their committed XML technology!



Bestsellers

1. XMLMate 2.0 from Insight Software \$147.99
2. VisualAge for Java Professional Edition v3.0 from IBM \$124.99
3. Optimix 3.0 Professional from Intuitive Systems \$448.99
4. Meta 1P v4.1 Suite Standard 1000 from Checkpoint \$3799.99
5. NetMax File Server from Cybernet Systems \$73.99

What's Online

www.sys-con.com/xml

XML-Journal.com

Can't get to the newsstands on time? Let www.xml-journal.com be your source for industry events and happenings. Check in every day for up-to-the-minute news, events, and developments, and be the first to know what's going on in the industry.

Special Offers

Check out this area for free downloads from the industry's leading XML vendors, free Web seminars...you can even register for free giveaways! Just type in www.xml-journal.com.

JDJ Store CD Special

The complete library of *XML-J*, *JDJ*, and *CFDJ*, articles are available on CD at a special price, for a limited time.

Order now and have over a thousand articles on hand for research and review...features, how-tos, product reviews, case studies, tips & tricks, interviews, IMHOs, and more!

Check out over 150 XML articles covering topics such as XML in Transit, XML & B2B, Java & XML, The XML Files, XML & WML, VoiceXML, SYS-CON Radio, XML & XSLT, XML & XSL, XML & XHTML, 2B or Not 2B, XML Industry Insider, XML Script, XML & Business, XML Demystified, XML & E-Commerce, XML Middleware, and much more.

This exclusive package normally sells for \$260, but it can now be yours for only \$175.99. Order today and save!

Digital Edition

Don't have your print edition on hand? Can't wait for the next issue to arrive in the mail? Our digital edition is just what you need. As long as you have your computer with you, you can read *XML-Journal* anytime, anywhere.

Looking to research a specific topic? Search our archives - we've got every article that's been published since our premier issue!

Subscribe to Our Free Weekly Newsletters

Now you can have the latest industry news delivered to you every week. SYS-CON newsletters are the easiest way to keep ahead of the pack. Register for your *free* newsletter today! There's one for Java, XML, Web Services, Wireless, and ColdFusion. Choose one, or choose them all!

XML Forum

Join XML-JList, the new XML mailing list community. Join other IT professionals, industry gurus, and *XML-Journal* writers for XML discussions, technical questions, and more. Voice your opinions and assessments on topical issues or hear what others have to say.

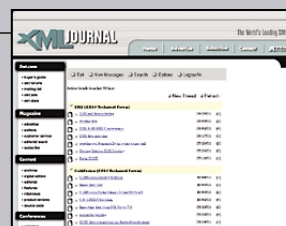
Join the XML-JList now to monitor the pulse of the XML industry! ☺



Subscribe to Our FREE Weekly Newsletters

☐ Java
☐ XML
☐ Web Services
☐ Wireless
☐ ColdFusion
☐ WebSphere
☐ WebLogic
☐ Linux

E-Mail Address:



QUICK POLL

Are you working with Web Services?

- ☐ A- Yes
☐ B- No

Results

xml spy
definitely stirred!

v3.5
FREE DOWNLOAD

Click for a
FREE 30-day trial
AtoWeb

Automatically
GENERATE
XSLT
INDUSLOGIC

carrig
learning

SoftQuad
XMetal
Free
Download

CAPE CLEAR

VoiceGenie
TECHNOLOGIES INC.



A jump start on the next generation of e-commerce

XMLTransformationTechnology forE-Business

Depending on which research you subscribe to, the consensus is that B2B e-business transactions are going to explode over the next few years. Forrester Research predicts \$2.7 trillion in the U.S. by 2003. GartnerGroup expects \$3.7 trillion for 2004, while the Boston Consulting Group, the most bullish, anticipates \$7 trillion worth of B2B transactions for the same year.

This growth isn't an isolated North American phenomenon. Many similar studies point to rapidly growing e-commerce trends in Europe, Asia, and Latin America. To get a better understanding of what's driving this growth, consider the following economic realities faced by today's businesses:

- A power company needs to rely on 20 other providers in the power grid to maintain service during peak loads, whereas once the company generated all of its own power.
- A manufacturer of PCs distributes product through 27,000 resellers over the Web, and the number is growing.
- A market aggregator runs a Web portal with no physical inventory of its own, but uses its branding to sell other distributors' products and attract customers to that overall branding.
- A car manufacturer has some 5,000 second-tier suppliers of parts, yet needs to track the flow of product, keep inventory levels to a minimum, and still maintain quality.
- A Web-based travel service aggregates offerings from 500 vacation package providers.
- An insurance company works through 20,000 field agents who need to operate one-on-one with clients.
- An office supply company has a customer list of 50,000 small businesses that order products over the Web, from printed catalogs, and over the telephone, and still demand next-day delivery.

Managing growth of this magnitude imposes enormous pressures on any

organization's IT resources. The sheer volume of information, coupled with an increasingly complex business model in which the distinction between client, supplier, partner, and competitor is no longer as clear as it once was, poses new and exciting challenges for IT professionals. Adding to the pressure are the rising expectations of both management and end users that are often out of sync with the capabilities of existing information systems not suited to today's electronic business climate. EDI (electronic data interchange) has been around for almost a quarter of a century, but with the growing demands of today's global economy it's becoming less relevant as a means of doing business electronically.

Standing directly in the path of the e-business juggernaut is XML technology, heralded as the next revolution in computing. Microsoft and others have staked their corporate futures on XML, believing it will be the foundation for true interoperability, allowing seamless electronic access to the global economy without being cost-prohibitive. In a recent interview with eCompany Now, Bill Gates said Microsoft is betting the company on XML and what it can do for businesses and consumers. He believes XML has the potential to be the glue that will allow developers to build Internet-native distributed applications.

For organizations that haven't yet dipped their toes into the XML pool, the excitement over the coming revolution is all well and good, but it begs a fundamental question. Where do you start? To answer this we need to go to the beginning and determine the nature of XML.

Simply put, XML is about data and a specific way to mark up or tag data. XML is a generic markup language that isn't concerned with how a document looks, but only with the document's content or data. In a sense XML frees information from proprietary data formats and makes it usable forever. By creating a document using XML, or by transferring a document into XML, you have data that will be understandable long after the originating application has become obsolete.

The potential impact of XML on e-business is clear. If all participants in the global economy, including SMEs (small and medium enterprises), are using a data format that's universally understandable by all and independent of proprietary applications, then business processes such as supply chain management, inventory control, and production tracking will become far more efficient. This will result in optimal resource allocation benefiting both producers and consumers.

As a case study, let's take a look at the U.S. health-care industry. According to Efficient Healthcare Consumer Response (www.csc.com/newsandevents/news/1227.shtml), supply chain costs of distributing medical products total approximately \$23 billion per year, of which an estimated \$11 billion could be eliminated by more efficient sharing of information, order management, and streamlined movement of products. In particular, the study highlights the need for, and the opportunity to reduce the heavy reliance on, telephones, faxes, catalogs, and electronic data interchange.

One company, MedChannel (www.medchannel.com), based in San Francisco and partially funded by Johnson & Johnson and Goldman Sachs, focuses on eliminating waste and increasing efficiency for all participants in the global

AUTHOR BIOS

C. Matthew MacKenzie is cofounder and vice president of R&D at XML Global Technologies, Inc. A former freelance consultant for e-commerce integration projects, Matthew served as chief architect of GoXML Search, created the first prototype for this product, and led all the research and development work.

Anthony Dutton is director of corporate communications at XML Global Technologies, Inc. Previously he worked with a corporate communications and public relations consultancy, representing a wide range of high tech firms.

health-care supply chain. As an example of the inefficiencies of the current system, MedChannel estimates annual cost savings on the order of \$2.7 billion could be achieved simply by eliminating order errors that ensue from incorrect data entry and improper data management. MedChannel's solution is the Contract Price Management Network, an Internet-based hub connecting all the parties involved in the purchase, sale, and distribution of health-care supplies and pharmaceutical products. By allowing collaboration among the various stakeholders, effective maintenance, measurement, and transmission of contractual information is possible.

To facilitate the Network, however, MedChannel needed to offer a technical solution allowing the universal sharing of information using a single data format. With many Network partners being large health-care providers, wholesalers and distributors, manufacturers, and group purchasing organizations, and with many using EDI and legacy systems, an XML transformation tool was essential. MedChannel elected to use GoXML Transform, an any-to-any transformation engine using Java enterprise portability, developed by XML Global Technologies, Inc. (www.xmlglobal.com). Using GoXML Transform will allow the MedChannel Network to facilitate low-cost, real-time Internet transmission of contract and price data, reducing or eliminating expensive dedicated electronic transmissions.

XML Transformation Options

Three fundamental approaches can be taken to accomplish transformation to and from an XML format:

- XSLT
- Custom programming/scripting
- Rules-based transformation

XSLT

XSLT is a template-based stylesheet language that's well suited for situations in which both the source and target format are XML. It's possible to handle other source and target formats by writing custom functions that can be "plugged" into an XSLT processor. The syntax used by XSLT is very much like any typical programming language, with keywords for loop control and control flow as well as conditionals and variables. The custom functions that can be written to extend XSLT's functionality require that the programmer constructing them fully understand the ins and outs of XSLT.

XSLT's syntax is like that of a programming language, with keywords that a programmer is comfortable with as its basis. A number of good XSLT editors in the

market make XML->XML/HTML more accessible, but none address making transformations from different formats – such as EDI – easier. XSLT syntax is at its worst when transforming non-XML content because this is where the programmer-oriented syntax comes into its own, the result being much more complex than a custom-scripted application in Perl.

Custom Programming and Scripting

Custom programming and scripting is the traditional approach in which code (using, for example, Java or C++) is written to read in source content, interpret its structure and semantic meaning, and form a new document with a different structure. This is commonly seen in Web-based database applications in which data from tables and rows is transformed into Web page content. Some vendors provide products that act as a programmer's transformation toolkit, providing common tools that ease the task of writing custom transformation code.

The custom programming and scripting approach to transformation can often provide very good performance, as it's a custom application, but it comes at the cost of flexibility, scalability, maintainability, and, for many organizations, feasibility. For most organizations this approach is often too expensive for all but the most specialized application. The custom programming approach is sometimes the best choice, but most of the time it isn't for reasons of maintainability and the tendency of these applications to become convoluted (spaghetti code).

Rules-Based Transformation

Rules-based transformation approaches the process from a non-programming perspective, using a series of assertions (or rules) to perform a transformation. To create a mapping between two structured formats, the user composes rules that define the relationships between the input and output structure.

For example, suppose your input database has two database columns: column A is "First Name" and column B is "Last Name." Suppose you want to create an XML document that has the element "Full Name." Using the rules-based approach you'd construct a rule similar to this: Output::FullName = DB::FirstName + DB::LastName. These rules are then typically threaded together in XML or a simple scripting language to control the execution flow and provide a serialization mechanism.

The rules-based approach is flexible and requires little programming knowledge. It empowers business domain experts to perform most transformation

SUBSCRIBE AND SAVE

XML JOURNAL

Offer subject to change without notice

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

DON'T MISS AN ISSUE!

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

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

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

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

REGISTER NOW. LIMITED SEATING



SAVE 30% off*
the annual
newsstand rate

JAVA DEVELOPER'S JOURNAL

*Offer subject to change without notice

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

DON'T MISS AN ISSUE!

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

In November JDJ:

Java Developer's Journal Readers' Choice Awards
A synopsis of the 23 winners and 69 finalists

Strutting Your Stuff
Create sophisticated applications
by Neal Ford

Add Fine Grained Access Control for Methods
Use the contents of a trace to find out information about the caller
by Ashutosh Narhari

J2ME Game Architecture
An introduction to game development for MIDP handsets
by Mark Quinn and Julien Tranchant

WEB SERVICES EDGE 2001 SANTA CLARA, CA



tasks. Since the syntax is declarative and not programmatic, all that's needed is a thorough understanding of the input and output formats. Like XSLT, the rules can be changed easily as formats require changing, but unlike XSLT and custom programming, rules-based transformation doesn't require the user to do any programming or understand basic programming concepts (although such knowledge is helpful when looking at raw business transaction data). Rules-based transformation typically is format agnostic as well. Rules-based transformation products, such as GoXML Transform, are generally composed of a core transformation engine, an input data parser (which by default understands many common input formats such as JDBC, XML, EDI, and CSV), and a graphical user interface allowing the user to create mapping rules through a visual interface. While the rules-based transformation methodology's guiding principle is that transformation tasks shouldn't require a programmer, a good rules-based transformation product will make all of its input and output APIs available so that, with the help of a programmer, virtually any data format that has structure can be transformed.

If implemented in the correct fashion, the rules-based approach is an ideal way to handle XML transformation, and – by way of its architecture – any-to-any transformations. By organizing the transformation into distinct areas such as input, output, and rules, an enterprise can begin to see enormous cuts to the bottom line of associated development in the exact same transformation attempted by other methodologies. By using scripting as the glue to bring together many of the individual components of a rules-based system, you bring about a much more robust and cost-effective solution to your transformation projects. This is the ideal method of business analyst-driven XML and legacy data transformation.

Use cases for each methodology could be similar to the following. With XSLT, a company may already have existing XML and need to integrate with a partner who has XML as well, but a different flavor. XSLT would enable the transformation of XML from one format to another.

For the custom programming and scripting approach, you'd have a large enterprise (no other size could really afford all the customized programming work for this), which requires transformations to occur, most likely on a project-to-project basis and not in an ongoing partnership arrangement of data interchange. They would hire a company to come in and spend weeks creating programs to accomplish this. These pro-

grams are reusable only in the exact same context as the original purpose. Further needs dictate further expenditures and associated delays in bringing in programmers to belt out more programs or fix old ones that have stopped working.

Rules-based transformations would be enjoyed by practically any size company and designed for reuse if built correctly. Companies would have cheaper original implementation, a modest maintenance requirement (as all templates are reusable by simply changing the input/output), and the ability to bring in their requirements for business process inclusion. All this can be accomplished by using business analysts in their data-domain expert roles, not your developers – who should be occupied with other revenue-generating activities – or highly paid external consultants.

If you examine the various choices associated with each methodology, you can quickly discern the obvious choice for you. Our product, GoXML Transform, includes a patented algorithm that efficiently allows the input, output, and rules to be separated and treated like individual components. This lets the enterprise dictate the inputs and outputs without changing the actual mappings. This is extremely powerful and grants a much more cost-efficient deployment and ongoing maintenance of your data transformation needs. There's no need to duplicate mapping work such as in custom programming or scripting, and there's much more flexibility and business process than XSLT.

As discussed earlier, XML transformation technology is of fundamental importance to a firm looking to migrate business applications to the Internet using an XML platform. Quite apart from the obvious advantages of using XML and thereby being able to participate in an open and accessible environment, XML transformation technology allows organizations to leverage their existing information assets. Kate Fessenden of the Aberdeen Group has suggested that by using XML transformation technology, firms can breathe new life into their EDI assets without having to commit to costly and time-consuming IT expenditures. By deploying XML transformation technology, new participants in the Internet economy, as well as those who have a 25-year investment in EDI, will be able to generate smart data, allowing many-to-many and machine-to-machine transactions that will pave the way for the next generation of e-commerce. ☎

MATT @ XMLGLOBAL.COM

ANTHONY.DUTTON @ XMLGLOBAL.COM


XML-J ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
ALTOVA	www.xmlspy.com		21, 68
CAPE CLEAR	www.capedclear.com/ca30	866-CAPE226	13
DATA MIRROR	www.datamirror.com	800-362-5955	6
HIT SOFTWARE	www.hitsw.com	408-345-4001	15
IONA TECHNOLOGIES	www.iona.com/ws-webcasts		2
IXIASOFT	www.ixiasoft.com		17
JAVA DEVELOPER'S JOURNAL	www.sys-con.com	800-513-7111	57
JDJSTORE.COM	www.jdjstore.com	888-303-JAVA	29, 36-37
OMNIMARK TECHNOLOGIES	www.omnimark.com	613-745-4242	9
ROGUE WAVE SOFTWARE	www.roguewave.com/guru2		4
SOFTWARE AG	www.softwareagusa.com/xmljac	877-SAG4XML, ext 550	67
SYS-CON MEDIA, INC. REPRINTS	www.sys-con.com	201-802-3026	25
SYS-CON EVENTS	www.sys-con.com	201-802-3069	25
WEB LOGIC DEVELOPER'S JOURNAL	www.sys-con.com/weblogic	800-513-7111	49
WEB SERVICES JOURNAL	www.sys-con.com/webservices	800-513-7111	60
WEB SPHERE DEVELOPER'S JOURNAL	www.webspheredevelopersjournal.com	800-513-7111	49
WIRELESS BUSINESS & TECHNOLOGY	www.sys-con.com	800-513-7111	59
WIRELESS EDGE CONFERENCE & EXPO	www.sys-con.com	201-802-3069	41
X-HIVE	www.x-hive.com		25
XML JOURNAL	www.sys-con.com	800-513-7111	55
XML TRAINING CALENDAR			63
XML TRAINING COURSES			62



www.wbt2.com

SUBSCRIBE NOW


www.javadevelopersjournal.com



www.xml-journal.com


TO THE

www.coldfusionjournal.com




FINEST

www.powerbuilderjournal.com




TECHNICAL

www.webspheredevelopersjournal.com




JOURNALS

www.wldj.com



IN THE

www.wsj2.com



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

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

Next Month...

The Next-Gen User Interface
How will Sun ONE and Microsoft .NET applications be presented to end users?
Your guess is as good as theirs, but revolutionary ideas are emerging.
by Russ Atkind and Sean Harvey

XSL Extensions and Java
What are XSLT extensions?
by Roy Hoobler

XML and Security
Some of the initiatives that will allow XML to travel safely between enterprises...
by Mark O'Neill

Got XSLT? - Part 3
How to transform XML to VoiceXML easily - and with style
by Shouki Souri

Dynamic Multimedia Presentations Using XSLT
A virtual tour slide show application combining SMIL implementation and XSLT transformation
by Siet Leng Lai



XML JOURNAL

Don't miss the December issue!



Using XML and XSL transformation to create a WAP-compatible WML file

GotXSLT?

Part 2 of 3

XSLT is a powerful technology that can provide many benefits. In this tutorial we examine how you can use XSLT to transform an XML example to the WML (Wireless Markup Language) that WAP (Wireless Application Protocol) devices such as PDAs – Palm Pilots, digital cell phones, pagers – use or understand.

In Part 1 of this series (Vol. 2, issue 10) we briefly discussed how to transform our XML document into HTML by using Microsoft's Internet Explorer (MSXML parser 2.0). We were able to do this because IE 5.0 is more than just an XML parser – it's also an XSLT processor. This kind of transformation is sometimes called *client-side transformation*, that is, the client (IE 5.0) performed the transformation.

In the case of WAP, the WAP devices can't perform the transformation since they're typically "dump" devices with limited processing power and memory (at least for the time being). Therefore, in a real-world case, the transformation is most often performed on the server side. In other words, the Web server transforms the XML document into WML.

This tutorial requires the following software:

- LotusXSL from www.alphaworks.ibm.com/tech/LotusXSL
- Openwave UP.SDK, release 4.0 or 4.1, from <http://developer.openwave.com/>
- JDK 1.2 or 1.3 from www.javasoft.com (I assume you've already installed JVM on your machine)

LotusXSL implements an XSL processor in Java that can be used from the command line, in an applet or a servlet, or as a module in other programs. By default, it uses the XML4J XML parser (available as a xerces.jar file when you download LotusXSL) using xerces, but it can interface to any XML parser that conforms to the DOM level 2 or SAX level 1 specification. The freely available LotusXSL used to be a tool developed by alphaWorks, but has now been donated to the Apache project, which maintains it under the

name Xalan. LotusXSL is a version (or distribution) of Xalan that has been tested and approved by IBM alphaWorks.

Openwave's software developer kit, UP.SDK, is the leading solution for creating WAP 1.1-compatible Internet services and applications for wireless handsets. The freely available UP.SDK contains the tools and interfaces necessary for a developer to create innovative applications for the wireless Internet.

UP.SDK includes the robust UP-Simulator software, available for Windows platforms and designed to accurately simulate the behavior of a variety of Internet-enabled handsets.

Finally, you need a JVM since LotusXSL is a Java implementation of the XSL engine and needs a container to run.

• • •

The primary purpose of this tutorial is to introduce you to the following concepts:

- Transforming an XML document into WML
- Using XSLT, XML, and WML

What Is WAP?

WAP – Wireless Application Protocol – is the leading standard for information services on wireless terminals like digital mobile phones. The WAP standard is based on Internet standards (HTML, XML, and TCP/IP). It consists of a WML language specification, a WMLScript specification, and a Wireless Telephony Application Interface (WTAI) specification.

WAP is published by the WAP Forum, founded in 1997 by Ericsson, Motorola, Nokia, and Unwired Planet. Forum members now represent over 90% of the global handset market, as well as leading infrastructure providers, software developers, and other organizations.

What Is WML?

WML – Wireless Markup Language – is a markup language inherited from HTML, but it's based on XML, so it's much stricter than HTML.

WML is used to create pages that can be displayed in a WAP browser. Pages in WML are called *decks*, which are constructed as a set of *cards*. WML is mostly about text. Tags that would slow down the communication with handheld devices are not a part of the WML standard, and the use of tables and images is strongly restricted.

Since WML is an XML application, all tags are case sensitive (<wml> isn't the same as <WML>), and all tags must be properly closed.

When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. Navigation between the cards is done by the phone computer (inside the phone) without any extra access trips to the server. See the **Resources** section for more information about WAP and WML.

Okay, enough theory stuff...let's get started!

Recalling Our XML Document

I hope you still remember our XML example (mybooks.xml). To refresh your memory, Listing 1 depicts our XML file. As explained in my previous article, you're free to modify your XML document any way you like, but you'll then have to pay extra attention to some of the details when we get into creating our WML stylesheet.

Note that we won't reference any XSL stylesheet onto our XML document. This is because we'll explicitly plug in our WML, XSL, and XML documents to perform the translation from a command line.

Our goal is to produce a WML page that looks like Figure 1, using UP.SDK simulator software.

AUTHOR BIO

Shouki Sourj, a lead software engineer at PanAmSat Corporation, is experienced in Java, CORBA, XML/XSL, C/C++, and other technologies. Shouki holds a bachelor's degree in electrical engineering and a master's degree in computer science.

Creating the Stylesheet Document

Let's see how we can devise a stylesheet to display the mybooks.xml document nicely on a WML browser. Listing 2 shows a complete XSL file to render our XML file (mybooks.xml) into a WML page. As you can see, it's a bit more complex than our previous XSL document.

Understanding the WML Stylesheet

Let's now take a closer look at our mybooks_wml.xsl stylesheet. The second line of the XSLT stylesheet is:

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

Here we define the namespace for the XSL stylesheet with the `<xsl:stylesheet>` element. Note that we use the new W3C XSLT recommendation, which requires an additional attribute named *version* to be included. In the previous article we used MSXML 2.0, which is not 100% compatible with this recommendation, so we use the following line instead:

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

The `<xsl:output>` specifies options for use in serializing the result tree. In this case we output nicely indented XML:

```
<xsl:output method="xml" indent="yes"/>
```

The `<xsl:template>` element:

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

is used as a template to match against the source XML document. This is a template that corresponds to the root (/) of the XML source document.

Now we begin to construct the body of our WML page using `<wml>` tags:

```
<wml>
<xsl:apply-templates select="//ITEM" />
</wml>
```

The `<xsl:apply-templates>` element first selects a set of nodes using the query specified in the *select* attribute. If this attribute is left unspecified, all children of the current node are selected. For each of the selected nodes, `<xsl:apply-templates>` directs the XSLT processor to find an appropriate `<xsl:template>` to apply. In our case the XSLT processor will apply all ITEM templates (see Listing 3).

Now we'll start to define those ITEM templates. The `<xsl:element>` element allows an element to be created with a

computed name. The name of the element to be created is specified by a required name attribute (*card*) and an optional namespace attribute. In other words, this creates the `<card>` element with attributes *id*="#Book<xsl:number/>" and *title*="My Book Catalog". The element `<xsl:number/>` is an empty element and serves as a counter.

Similarly, we created the element `<do>` with the attribute *type*="accept" and the label="Next Book".

The last element we created is `<go>` with the attribute name *href*="#Book<xsl:number value="position() + 1". This translates to #Book2 (since the position of this template is 1).

We close the card and do elements.

The XSL `<xsl:value-of>` element can be used to select the value of the element matched by the *select* predicate.

The XSL stylesheet then closes all open element tags in order, with no nesting allowed.

Listing 4 is the second half of our stylesheet.

The XSL `<xsl:template match="ITEM[position() = last()]>` checks for the last ITEM element in our XML document. If it actually is the last element, the XSLT processor will continue with the next line; otherwise it will skip and apply `<xsl:template match="ITEM">` again. The rest is pretty simple, except for element *go* attribute `<xsl:attribute name="href">#Book<xsl:number value="count(//ITEM) - count(//ITEM) + 1"/>`.

The XSLT `count()` function returns the number of nodes in the argument node-set. In our example it returns three (since there are only three books in our XML document). Thus the result is an attribute value of #Book1, which forces the WML browser (or our WAP simulator) to go back to the Book1 card.

I'm sure there are several other and possibly easier ways to create this behavior using other XSLT functions, but I'll leave this as an exercise for the reader.

Performing Your Own Transformations

Both Xalan (LotusXSL) and UPSDK have an HTML start-up page: *readme.html* (make sure to read the getting-started link) and *upsdk40.html*, respectively. They will show you how to set your classpath for the Xalan processor. At a minimum, you must include *xalan.jar* and *xerces.jar* on your system class path to perform the translation.

Java.org.apache.xalan.xslt.Process provides a basic utility for performing transformations from the command line. The command line for most standard transformations is as follows:

SAVE 30% off*

the annual newsstand rate

BUSINESS & TECHNOLOGY
wireless

*Offer subject to change without notice

ANNUAL NEWSSTAND RATE

~~\$71.88~~

YOU PAY

\$49.99

YOU SAVE

30% Off the Newsstand Rate

DON'T MISS AN ISSUE!

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

In November WBT:

A WBT Special: "i-mode 101"

For all wireless operators and developers hoping to imitate NTT DoCoMo's wildly successful i-mode offering in Japan.

Is Wireless Broadband Barreling Your Way?

As wireless broadband connectivity spreads around the globe, what technologies are taking us there?

Wireless ROI

Are you getting what you paid for? Asks the CEO of Cotelligent

Location is King

The president & CEO of go2 Systems offers his 35,000-ft view on location-based services.

Wireless Primer

SAVE THE DATE! WIRELESS EDGE MAY 7-9, 2000



SUBSCRIBE AND SAVE

WebServices JOURNAL

Offer subject to change without notice

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

DON'T MISS AN ISSUE!

Receive 12 issues of *Web Services Journal* for only \$69.99! That's a savings of \$13.89 off the annual newsstand rate.

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

In November WSJ:

Content Management & Web Services
eContent services — the services behind Web services

JAXM: Interoperable SOAP Communications for the Java Platform
Extensibility expedites e-commerce

Web Services: Enabling Technology or New Programming Paradigm?
The new integration model

The Depth and Breadth of the Business Opportunity
Revenue growth and improved efficiency



```
java org.apache.xalan.xslt.Process -
in xmlSource -xsl stylesheet -out
outputfile
```

where xmlSource is the XML source file name, stylesheet is the XSL stylesheet file name, and outputfile is the output file name. If you want the output to be displayed onscreen, simply omit the -out flag and argument.

Both mybooks.xml and mybooks_wml.xsl are saved under gotxslt directory as d:\gotxslt*.*

To perform the translation, type the following:

```
D:\>cd gotxslt
D:\gotxslt>java
org.apache.xalan.xslt.Process -in
mybooks.xml -xsl mybooks_wml.xsl -out
mybooks.wml
```

If all goes well, the new output file mybooks.wml will be the resultant output from the XML to WML translation (see Figure 2). The file mybooks.wml is a pure WML ASCII file, so you can open it and study it in any text editor.

Testing the Deck

In order to test our WML output file, you need to start the UP Simulator program (upsim.exe) and type the URL of your file (in this case file:///d:/gotxslt/mybooks.wml) in the Go field and press

Enter. The results of a successful request are shown in Figure 3:

Upon clicking the Next button (which is just below the text labeled "Next Book"), a second card will display the second book. Clicking again for our third book, you'll see that this time the label shows "Back" (see right-hand picture in Figure 3). Clicking on the Next button again will take you to our first book, and the loop will continue.

Conclusion

This article has shown a way to use XML and XSL transformation to create a WAP-compatible WML file. We've just scratched the surface of the potential of WAP/WML. XSLT is a powerful technology that can provide many benefits and hopefully you'll now feel confident to play with other examples on your own.

In Part 3 we'll translate our XML file into VoiceXML.

Resources

- WAP/WML forum: www.wapforum.org/
- LotusXSL: www.alphaworks.ibm.com/tech/LotusXSL
- Apache XML Project: <http://xml.apache.org/>
- Openwave UP.SDK Software Developer Kit, Release 4.0 or 4.1: <http://developer.openwave.com/>

SSOURI @ PANAMSAT.COM

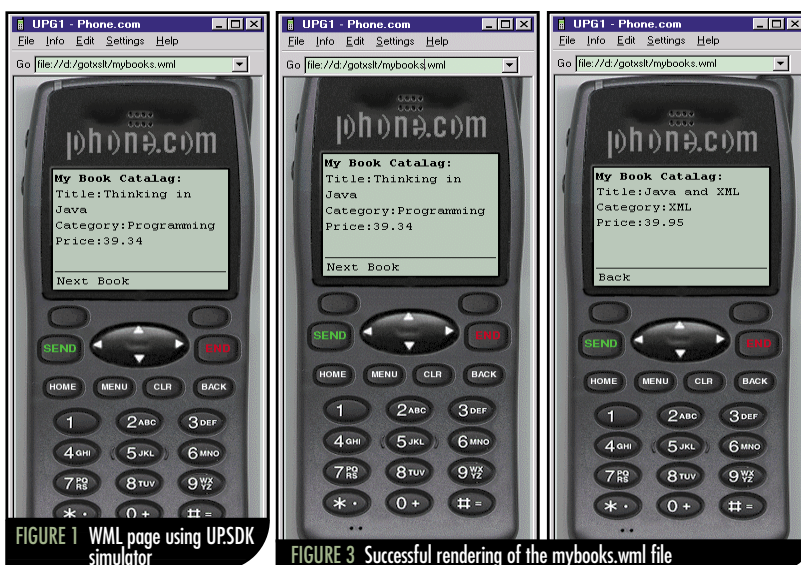


FIGURE 1 WML page using UP.SDK simulator

FIGURE 3 Successful rendering of the mybooks.wml file

```
Command Prompt
D:\gotxslt>
D:\gotxslt>
D:\gotxslt>
D:\gotxslt>
D:\gotxslt>java org.apache.xalan.xslt.Process -in mybooks.xml -xsl mybooks_wml.xsl -out mybooks
.wml
===== Parsing file:D:/gotxslt/mybooks_wml.xsl =====
Parse of file:D:/gotxslt/mybooks_wml.xsl took 620 milliseconds
Parse of mybooks.xml took 91 milliseconds
Transforming...
transform took 60 milliseconds
XSLProcessor: done
D:\gotxslt>
```

FIGURE 2 Completing the XML to WML transformation using the Xalan processor

LISTING 1 XML document from Part 1 tutorial

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!-- Disregard next statement for now -->

<BOOKS>
  <BOOK>
    <ISBN>1-565-92869-9</ISBN>
    <CATEGORY>Programming</CATEGORY>
    <RELEASE_DATE>2000-03-21</RELEASE_DATE>
    <TITLE>Enterprise JAVABEANS</TITLE>
    <PRICE currency="US">34.95</PRICE>
  </BOOK>
  <BOOK>
    <ISBN>1-861-00311-0</ISBN>
    <CATEGORY>XML</CATEGORY>
    <RELEASE_DATE>2000-03-10</RELEASE_DATE>
    <TITLE>PROFESSIONAL XML</TITLE>
    <PRICE currency="US">49.99</PRICE>
  </BOOK>
  <BOOK>
    <ISBN>0-596-00016-2</ISBN>
    <CATEGORY>XML</CATEGORY>
    <RELEASE_DATE>2000-07-11</RELEASE_DATE>
    <TITLE>JAVA and XML</TITLE>
    <PRICE currency="US">39.95</PRICE>
  </BOOK>
</BOOKS>
```

LISTING 2 "mybooks_wml.xml": Complete XSL file to transform XML into WML

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/" />
</wml>
<xsl:apply-templates select="//ITEM" />
</wml>

</xsl:template>
<xsl:template match="ITEM">
  <xsl:element name="card">
    <xsl:attribute name="id">Book<xsl:number/></xsl:attribute>
    <xsl:attribute name="title">My Book
      Catalog</xsl:attribute>
  <xsl:element name="do">
    <xsl:attribute name="type">accept</xsl:attribute>
    <xsl:attribute name="label">Next Book</xsl:attribute>
  <xsl:element name="go">
    <xsl:attribute name="href">#Book<xsl:number value=
      "position() + 1"/></xsl:attribute>
  </xsl:element>
</xsl:element>
  <p>
    <big>My Book Catalog:</big>
    <br/>
    <xsl:text>Title:</xsl:text>
    <xsl:value-of select="TITLE"/>
    <br/>
    <xsl:text>Category:</xsl:text>
    <xsl:value-of select="CATEGORY"/>
    <br/>
    <xsl:text>Price:</xsl:text>
    <xsl:value-of select="PRICE"/>
  </p>
</xsl:element>
</xsl:template>
<xsl:template match="ITEM[position() = last()]>
  <xsl:element name="card">
    <xsl:attribute name=
      "id">Book<xsl:number/></xsl:attribute>
    <xsl:attribute name="title">My Book Catalog</xsl:attribute>
  <xsl:element name="do">
    <xsl:attribute name="type">accept</xsl:attribute>
    <xsl:attribute name="label">Back</xsl:attribute>
  <xsl:element name="go">
```

```
    <xsl:attribute name="href">#Book<xsl:number value=
      "count(//ITEM) - count(//ITEM) + 1"/></xsl:attribute>
  </xsl:element>
  <p>
    <big>My Book Catalog:</big>
    <br/>
    <xsl:text>Title:</xsl:text>
    <xsl:value-of select="TITLE"/>
    <br/>
    <xsl:text>Category:</xsl:text>
    <xsl:value-of select="CATEGORY"/>
    <br/>
    <xsl:text>Price:</xsl:text>
    <xsl:value-of select="PRICE"/>
  </p>
</xsl:element>
</xsl:template>
</xsl:stylesheet>
```

LISTING 3 <xsl:template match="ITEM">

```
<xsl:element name="card">
  <xsl:attribute name="id">Book<xsl:number/></xsl:attribute>
  <xsl:attribute name="title">My Book Catalog</xsl:attribute>
<xsl:element name="do">
  <xsl:attribute name="type">accept</xsl:attribute>
  <xsl:attribute name="label">Next Book</xsl:attribute>
<xsl:element name="go">
  <xsl:attribute name="href">#Book<xsl:number value=
    "position() + 1"/></xsl:attribute>
</xsl:element>
</xsl:element>
<p>
  <big>My Book Catalog:</big>
  <br/>
  <xsl:text>Title:</xsl:text>
  <xsl:value-of select="TITLE"/>
  <br/>
  <xsl:text>Category:</xsl:text>
  <xsl:value-of select="CATEGORY"/>
  <br/>
  <xsl:text>Price:</xsl:text>
  <xsl:value-of select="PRICE"/>
</p>
</xsl:element>
</xsl:template>
```

LISTING 4 <xsl:template match="ITEM[position() = last()]>

```
<xsl:element name="card">
  <xsl:attribute name= "id">Book<xsl:number/></xsl:attribute>
  <xsl:attribute name="title">My Book Catalog</xsl:attribute>
<xsl:element name="do">
  <xsl:attribute name="type">accept</xsl:attribute>
  <xsl:attribute name="label">Back</xsl:attribute>
  <xsl:element name="go">
    <xsl:attribute name="href">#Book<xsl:number value=
      "count(//ITEM) - count(//ITEM) + 1"/></xsl:attribute>
  </xsl:element>
</xsl:element>
  <p>
    <big>My Book Catalog:</big>
    <br/>
    <xsl:text>Title:</xsl:text>
    <xsl:value-of select="TITLE"/>
    <br/>
    <xsl:text>Category:</xsl:text>
    <xsl:value-of select="CATEGORY"/>
    <br/>
    <xsl:text>Price:</xsl:text>
    <xsl:value-of select="PRICE"/>
  </p>
</xsl:element>
</xsl:template>
</xsl:stylesheet>
```

XML Training

Crane Softwrights

Lecture and hands-on training materials in XML-related Recommendations can be licensed for both public and in-house use. See <http://www.CraneSoftwrights.com/xj/schedall.htm> for syllabi and deliveries by G. Ken Holman, <http://www.CraneSoftwrights.com/xj/sched.htm> for deliveries in Ottawa (including "train the trainer" for licensees), and

<http://www.CraneSoftwrights.com/xj/licensing.htm> for licensees who can deliver Crane's courses at your site.

www.CraneSoftwrights.com/xj
(613) 489-0999



DMSi - Document Management Solutions, Inc.

DMSi presents a full suite of solutions-based training services focused on the specific needs of your environment. Courses include data architecture development, XML, XSL/T and related standards, and product-specific training. We customize these courses to include your data and scenarios related to your organization. The result is a class that

satisfies the specific training requirements of your staff.

www.dmsi-world.com
(978) 738-9770



ISOGEN International

Acquire practical XML skills the way the professionals do, with either full track or individual classes based on the ISOGEN XML Learning Framework. Our pragmatic and highly refined approach, our world-class instructors, and our excellent materials guide you to a higher level of XML skill. Open enrollment, onsite training, and customized courses available.

www.isogen.com or www.datachannel.com
(612) 961-6203



OMNIMARK Technologies

OmniMark Technologies, in conjunction with Stilo Technology in the UK, offers a broad range of XML-related courses covering both design and processing. Courses are offered at OmniMark facilities in Boston, Paris, and Leuven, and at Stilo's facilities in London. Custom and on-site training are also available. For more information see our Web site.

www.omnimark.com or www.stilo.com
(613) 745-4242



Online-Learning.com

Offers online courses for professional development; learn XML, XSL, XSLT, Technical Writing, and Usability. All courses provide extensive instructor support, interaction, practical hands on experience, and certification.

Let us teach you how to deliver single source content.

www.online-learning.com/xmljournal.html or
info@online-learning.com (613) 596-5673

ONLINE-LEARNING

TrainingCity

TrainingCity provides XML Consulting services & hands on XML/XSLT training, workshops & elearning. Customized onsite training, CDROMs, & Videos. Call to speak with one of our XML Experts: 858-755-8868, <mailto:info@TrainingCity.com> info@TrainingCity.com, www.TrainingCity.com for course outlines & free online sample training!

www.TrainingCity.com
(858) 755-8868

TrainingCity

For listing opportunities contact
Megan Ring (201) 802-3023 megan@sys-con.com

November 2001

Monday	Tuesday	Wednesday	Thursday	Friday
<div>OmniMark: XML and Data Transformation, London</div> <div>5</div>	<div>6</div>	<div>7</div>	<div>8</div>	<div>9</div>
AppDev: XML and XSLT Instructor-Led Training, Level 1, Chicago, IL		AppDev: XML and XSLT Instructor-Led Training, Level 2, Chicago, IL		
Global Knowledge: XML Fundamentals, Columbus, OH		OmniMark: XML Processing with OmniMark, London		
Global Knowledge: XML Fundamentals, Orlando, FL				
	Global Knowledge: XML Fundamentals, Atlanta, GA			
DMSi: Introduction to XML and XSLT, Available onsite				
12	<div>OmniMark: XML Schema Message Development Workshop, London</div> <div>13</div>	<div>14</div>	<div>15</div>	<div>16</div>
AppDev: XML and XSLT Instructor-Led Training, Level 1, Washington, DC		AppDev: XML and XSLT Instructor-Led Training, Level 2, Washington, DC		
Global Knowledge: XML Fundamentals, Dallas, TX		OmniMark: XSLT Masterclass, London		
OmniMark: Converting to XML Using OmniMark, London			DMSi: SGML/XML Advanced Topics, Available onsite	
XMLMentor: XML Certification Fast Track Course, Montreal				
Online-Learning.com: Professional Technical Writing, Introduction to XML, Professional XML Authoring, XSL Techniques, Usability Testing, Information Design Forum Classes available online or on CD	Global Knowledge: XML Fundamentals, New York City			
19	20	21	22	23
OmniMark: Web Programming with OmniMark, London				
OmniMark: Data Modelling Masterclass, London		OmniMark: XML Programming Masterclass (for non-Java development), London		
26	27	28	29	30
OmniMark: XML Programming Masterclass (for Java Development), London				
OmniMark: XML Schema Message Development Workshop, Boston, MA	Global Knowledge: XML Fundamentals, Denver			
	OmniMark: XSLT Masterclass, Boston, MA			
	Global Knowledge: XML Fundamentals, Dulles, VA		OmniMark: Data Modelling Masterclass, Boston, MA	
AppDev: XML and XSLT Instructor-Led Training, Level 1, Atlanta, GA		AppDev: XML and XSLT Instructor-Led Training, Level 2, Atlanta, GA		
OmniMark: Integrating Information with OmniMark, London				

XML NEWS



Winners of 2001 *JDJ* Readers' Choice Awards Announced

The winners of the 2001 *JDJ* Readers' Choice Awards were announced at an awards ceremony held at SYS-CON Media's JDJEdge 2001—International Java Developer Conference & Expo on September 24 at the Hilton New York.

The *JDJ* Readers' Choice Awards program has become the most respected industry competition of its kind and is often referred to as the "Oscars of the software industry." Results of this year's voting appear in the November 2001 issue of *Java Developer's Journal*.

The number of *JDJ* readers who cast their votes this year doubled compared to last year, to over 40,000. The total of award nominations also reached a new high in 2001, with more than 180 companies nominating over 350 products in 23 award categories. Seven new categories, including Best Wireless Java Application and Best XML Tool, were added this year.

"We were thrilled with the number of responses we received," said Alan Williamson, editor-in-chief of *JDJ*. "More products were nominated than ever before, and the participation in the voting process was vigorous throughout the five-month period."

Award recipients are selected through reader-submitted nominations, followed by online voting. An independent research firm managed the voting process.

Following is a list of the winning companies and products:



Best Java Book: *Java in a Nutshell*, 3rd edition, by David Flanagan (*O'Reilly*)

Best Code Protection Tool:
CCC/Harvest (*Computer Associates*)

Oracle 9i Best Enterprise Database Product:
Oracle 9i Database (*Oracle*)

Best Embedded Java Application:
Cloudscape (*IBM*)



Best Java Application:
BEA WebLogic Server (*BEA Systems*)

Best Java Application Server:
BEA WebLogic Server (*BEA Systems*)

Best JavaBean or Component:
WebLogic Commerce Server (*BEA Systems*)

Best Java Class Library:
Host Access Class Library (*IBM*)

Best Java IDE: VisualAge for Java 3.5 (*IBM*)



Best Java Installation Tool:
MultiPlatform Edition (*InstallShield*)

Best Java Middleware:
WebSphere Host On-Demand 5 (*IBM*)

Best Java Modeling Tool:
Rational Rose 2001
(*Rational Software*)



Best Java Profiling Tool: Rational Quantify
(*Rational Software*)

Best Java Reporting Tool:
9iAS Discoverer (*Oracle*)



Best Java Testing Tool:
— TeamTest (*Rational Software*)

Best Java Training Tool:
Total e-Server/J2EE Tutorial Trail Map
(*HP Bluestone*)



Best Java Virtual Machine:
Java HotSpot Performance
Engine Architecture
(*Sun Microsystems*)



Best Mobile Database Product:
SQL Anywhere Studio 7.0 (*Sybase*)



Best Team Development Tool:
VisualAge for Java 3.5 (*IBM*)

Best Wireless Java Application:
WebSphere Transcoding
Publisher (*IBM*)



Best XML Tool: X-IT
(*IBM alphaWorks*)



Most Innovative Java Product:
VisualAge for Java 3.5 (*IBM*)

Best Java Messaging Tool: MQ Series (*IBM*)

A complete list of *JDJ* Readers' Choice Awards winners and finalists is available at:
www.sys-con.com/2001/PR/code.cfm?page=09242001a

NeoCore and Tactica Form Partnership

(*Colorado Springs, CO*) – NeoCore has announced formation of a partnership with Tactica. Under the terms of the agreement,

NeoCore and Tactica will jointly offer products and services to provide tailored solutions for their customers' XML information management needs.



www.neocore.com
www.tactica.com

Arbortext Releases Epic 4.2

(*Ann Arbor, MI*) – Arbortext, Inc., has introduced Epic version 4.2. This latest version supports collaboration through change tracking, inline editing of document components, and easier conversion of multiple-content formats to XML.



Epic 4.2 also enhances Arbortext's long-standing support of XML standards through an industry-first combination of full support for XSL by offering both XSLT and XSL-FO publishing engines, enabling organizations to build an automated multi-channel publishing system.

www.arbortext.com

JDJ Announces Record Circulation

(*Montvale, NJ*) – SYS-CON Media has announced that circulation of *Java Developer's Journal* has reached 104,254 copies with the June 2001 print edition, generating single-issue revenues of over \$1 million. *JDJ* also announced a six-month circulation of 97,325 – up from the previous six-month average of 89,652 – during the period ending June 2001. In addition, *JDJ* delivered a 236,715-copy monthly circulation of its digital edition with the September 2001 issue.

XML NEWS



"We're very pleased to see that *Java Developer's Journal* is serving the fast-growing Java developer community around the globe as the hands-

down leading source of print and online information, and its lead increases with every issue," reports Fuat Kircaali, founder and CEO of SYS-CON Media. "Our hardworking SYS-CON team is very happy to be part of a leading company that offers unmatched quality information products in print and online, and with our i-technology conferences."

www.sys-con.com

SYS-CON Events Moving to Javits Center in 2002

(Montvale, NJ) – SYS-CON Events, Inc., will relocate its East Coast events to the Jacob Javits Convention Center in New York City in 2002. Web Services Edge 2002 East–International Web Services Conference & Expo and JDJEdge 2002–International Java Developer Conference & Expo will take place June 24–27, 2002. SYS-CON's East Coast events will complement the Technology Exchange Week New York – TECHXNY – a weeklong showcase for the most dynamic business technology players and products in the world.



"We look forward to coming back to New York City for the third year in a row and plan to double our exhibit floor by moving to the Jacob Javits Convention Center," said Grisha Davida, vice president of business develop-

ment at SYS-CON Media. "This will allow us to host Java, XML, .Net, and all related technologies under our Web services canopy."



"We will open the Call for Papers for next year's conference on October 22, 2001," according to Cathy Walters, vice president of SYS-CON Events.

www.sys-con.com

SYS-CON Media, Group Intelligence in Alliance

(Montvale, NJ) – SYS-CON Media and Group Intelligence have announced a strategic alliance in support of the fastest-growing Web-infrastructure market, WebSphere.

The companies will jointly deliver a full range of print, event, online, and digital services, creating the first information resource and infrastructure (InfoStructure) of the WebSphere marketplace.

WebSphere Developer's Journal, the newest i-technology publication from SYS-CON Media, will be published monthly starting in November.

www.sys-con.com
www.groupintelligence.com

ActiveEducation Releases XML Expert Test

(Golden, CO) – ActiveEducation, Inc., publisher of e-learning for information technology, has released the industry's first vendor-neutral, high-stakes certification examination for XML.

SYS-CON Media Launches Web Services Journal



(Montvale, NJ) – SYS-CON Media has debuted its premier issue of *Web Services Journal*,

the first print and online magazine devoted exclusively to the newest and most pervasive computing paradigm since the arrival of the Web itself.

The premier issue features articles from some of the best-informed developers, analysts, and executives in the i-technology world. This collector's issue includes a "CEO Round Table Discussion" led by executives and visionaries like David Litwack, CEO of Silverstream, Barry Morris, CEO of Iona, David Clarke, CEO of Cape Clear, Simon Phipps of Sun Microsystems, and Tyler Jewell of BEA Systems, among others.

www.sys-con.com/webservices

Candidates for XML Expert certification may register at www.2test.com/index.jsp.

The exam will be administered in North America at 1,358



Prometric Test Centers.

www.ActiveEducation.com

XyEnterprise Releases New Content@ Package

(Reading, MA) – A new, out-of-the-box Content@ package geared specifically to simplify the production of product documentation for product documentation editorial workgroups has been released by XyEnterprise.

The Content@ Product Documentation Solution package has all the standard Content@ features, such as a fully documented API for UNIX and NT, workflow and metadata management.



It also includes a direct application link to XyEnterprise's XML Professional Publisher (XPP) software for automated publishing and personalized, print-on-demand output.

www.xyenterprise.com

Connotate Upgrades vTag

(New Brunswick, NJ) – Connotate Technologies has released a major upgrade of its Web data extraction environment. Built on its XML-by-example extraction rule generator, vTag 2.0 increases coverage of Web-based data and improves speed by a factor of 150. Version 2.0 extends vTag's current capabilities to include construction of XML relationships that span multiple linked pages, and it enables



automated location of user-specific data identified by parameters supplied at the time of retrieval.

www.connotate.com

VoiceGenie Releases Genie IDE

(Toronto) – Genie IDE, a suite for developing, managing, and deploying VoiceXML applications, has been launched by VoiceGenie Technologies



Inc. The new IDE's extensive feature set enables users at all levels to rapidly develop and deploy enterprise-grade voice applications. Capabilities include file editing, file and project management, code assistance, validation, and platform simulation.

www.voicegenie.com

XML Schemas: The 'New' DTDs



[REVIEWED BY BOB SWART]

AUTHOR BIO

Bob Swart (aka Dr. Bob at www.drbbob42.com) is an IT consultant for the Everest Delphi Oplossings Centrum (DOC) in The Netherlands. Together with Hubert A. Klein Ikkink (aka Mr. Haki), Bob maintains Mr. Haki's JBuilder Machine (www.drbbob42.com/JBuilder), a Web site devoted to JBuilder.

DRBOB @ CHELLO.NL

Professional XML Schemas

by Jon Duckett, Oliver Griffin, Stephen Mohr, Francis Norton, Ian Stokes-Rees, Kevin Williams, Kurt Cagle, Nikola Ozu, Jeni Tennison

Published by WROX Press
Price: \$49.99

This book has nine authors (eight "real," since Kurt Cagle is mentioned as a contributing author), but only six photo faces appear on the cover. It doesn't really matter, as with many WROX "Professional" books, you're used to seeing a number of authors – usually experts in the field – covering different topics and subjects all related – in this case – to XML Schemas.

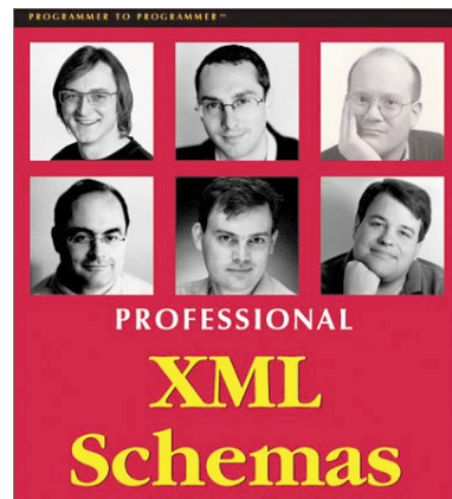
Is it possible to write an entire book about the topic of XML Schemas? Yes, it is. And the book makes this clear in the very first chapter, which explains that XML Schemas are in fact the next-generation DTDs. In fact, the introduction of the book mentions the "severe limitations" of DTDs and the fact that the W3C introduced XML Schemas as their replacement.

Having read the book, I must say that it certainly helps if you have some prior understanding of XML and the "old" DTDs. The book (only) refers to DTD, but describes the XML Schema language (and practical use) in a lot of detail – 16 chapters, to be specific, and five (an unusually low number for WROX Professional books) appendixes.

The first nine chapters cover the fundamentals of the XML Schema language in detail, including simple and complex datatypes, a validating parser (XSV), content models, deriving your own custom datatypes, the three different XML Schema namespaces, and so on. After reading these chapters, you should be able to understand XML Schemas as well as write your own to validate your XML documents.

The tenth chapter covers Schemas and XSLT, showing how we can use the latter to transform XML Schemas (which are still XML documents) and generate some useful documentation in HTML format. Quite useful actually. Chapter 10 and the remaining chapters in the book are more practically oriented, and cover potential real-world situations in which XML Schemas can be applied.

Chapter 10 is about XML Schemas and XSLT. Like me (and you, probably), the authors start by stating that it may be a bit odd to have a chapter on XSLT (transforming XML documents) in a book about XML Schemas (describing an XML document). However, if you think about it, an XML Schema itself is just an XML document as well, and using XSLT we can transform an XML Schema to something else. The chapter shows, for example, how to generate (useful) documentation from an XML Schema. To do this we must learn how to navigate through an XML Schema (in order to pull information from it). Another example that I'll be using myself someday is



how to create an HTML form from an XML Schema, using an XML document (based on the XML Schema) to populate the form. In the chapter summary we're again reminded of just one of the advantages that XML Schemas have over DTDs: the former are written in XML, which made this very interesting chapter possible in the first place.

Chapter 12 reminded me of the book *Professional XML Databases*, also published by WROX. It showed how to transform a Database Schema into an XML Schema. Given my background, I especially enjoyed reading the e-commerce case study in which business partners of a bank used XML Schemas to determine the correct data to be communicated (using SOAP, which made the chapter even more interesting to read, in my view). This chapter may be a foretaste of WROX's *Professional XML Web Services*.

Each chapter starts with an introduction (about one page) and ends with a summary (also about one page) that describes what you can expect to learn (or have learned). This helps you determine which chapters can be skipped or read later – especially in the second half of the book, the more practical chapters that show XML Schemas in a working context.

The appendixes range from an overview of Schema elements and attributes to a UML reference and a useful list of XML Schema tools.

This gets me to the only minor problem with the book: it could have used a CD with some of the examples and tools that appear in the book. Fortunately, the last appendix provides a bibliography and further reading, including a number of Web sites to investigate. And not to forget the XML section of WROX's own Web site, where you can download the source code (155KB).

If you've read *Professional XML* or *Professional XML Databases* (also from WROX – the latter was reviewed in *XML-J*, Vol. 2, issue 6), then this is a great follow-up book. If you know little about XML, you should first learn the basics (including some knowledge of DTDs) before you attempt to master XML Schemas with this book. Nevertheless, any serious state-of-the-art XML developer will have to learn about XML Schemas someday, and with that in mind I fully recommend this book. ☺

Software AG

www.softwareagusa.com/xmljac

Altova

www.xmlspy.com