

XML JOURNAL

The World's Leading XML Resource

Volume: 3 Issue:3

XML-JOURNAL.COM

XMLEdge
conference & expo


Page 29

JUNE 24-27
NEW YORK, NY

FROM THE EDITOR
XML Development
by Ajit Sagar pg. 5

INDUSTRY COMMENTARY
The Problem with XML-Based Storage
by Israel Hilerio pg. 7

BOOK REVIEW
XML's Contribution to Web Services
Reviewed by Bob Swart pg. 8


Happy 4th Birthday, XML!
pg. 34

AND NOW...
XML Is a Treasure My Heart Cannot Deny
by Tod Emko pg. 62

SYS-CON MEDIA



XML
MESSAGING
WITH JAXM

WRITTEN BY Mike Jastowski

How to exchange SOAP messages

PART 1 OF 2 10

Messaging: The World of Enterprise Messaging JP Morgenthal 16
A company's messaging infrastructure can reveal its IT smarts

XML Feature: XML Glue An XML workflow and integration layer for telecommunication providers Ron Ben-Natan 20

Publishing: Trends in High Volume XML Publishing Automated markup tools and techniques Evan Huang 26

Contrary Opinion: Markup is Madness Steve Klingsporn 32
The only real standard that's mandatory is TCP/IP – that aside, may the best new standards win!

Compression Techniques: XML Without Wires Mark O'Neill 36
PART 1 of 2 *Tailoring an XML-aware system to the wireless world*

XML Feature: Ant: An Introduction by Example A build tool based on XML, using Java classes Ann Black & Yu Tang 40

Testing Tools: A Client for Testing Server-Side XML Applications Help in speeding up the building and debugging process Jon Strande 48

XML Tutorial: Transforming XML Documents into HTML Stylesheets separate content from presentation Steve Hoenisch 52

Sonic Software Corporation

www.sonicsoftware.com

Sonic Software Corporation

www.sonicsoftware.com

Macromedia

www.macromedia.com/downloads

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@kimbo.com
SIMEON SIMEONOV simeons@macromedia.com

DEPARTMENT EDITORS

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

WRITERS IN THIS ISSUE

Ron Ben-Natan, Ann Black, Tod Emko, Israel Hilerio,
Steve Hoenisch, Evan Huang, Mike Jasnowski,
Steve Klingsporn, JP Morgenthal, Mark O'Neill, Ajit Sagar,
Jon Strande, Bob Swart, Yu Tang

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

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

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

EDITORIAL OFFICES

SYS-CON MEDIA

135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637
XML-JOURNAL (ISSN# 1534-9780)
is published monthly (12 times a year)
by SYS-CON Publications, Inc.
Periodicals postage pending
Montvale, NJ 07645 and additional mailing offices.
POSTMASTER: Send address changes to:
XML-JOURNAL, SYS-CON Publications, Inc.,
135 Chestnut Ridge Road, Montvale, NJ 07645.

©COPYRIGHT

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

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



WRITTEN BY AJIT SAGAR EDITOR-IN-CHIEF

from
the
editor

XMLDevelopment

Have you ever been called an XML developer or XML programmer? While processing XML in programs has become a common task for computer programmers, XML is used only in the context of some other programming language environment, such as Java, C++, and Web programming languages. We've all become accustomed to the presence of XML in the very guts of our applications over the last couple of years. Most IDEs in the market today offer tools for manipulating XML for the purpose of creating and deploying software components. Eventually the data formats expressed by XML have to be processed by the software components that either consume or produce the actual data.

XML IDEs have been evolving even as the standards have stabilized around XML. Existing programming IDEs such as Microsoft's Visual Studio .NET, Symantec's VisualCafé, Borland's JBuilder, and Macromedia's ColdFusion have all expanded their offerings to include XML processing in some form or another. For J2EE IDEs XML processing needs to be handled at least in deployment descriptors for Java components. With the advent of Web services, all these IDEs are now offering tools for converting J2EE components into Web services.

On the other hand, new vendors that exist for the primary purpose of processing XML have emerged in the last few years and have defined an XML-centric market of their own. They started out a few years ago offering parsing and editing features for XML documents. However, the IDE market for XML has expanded in conjunction with the expansion of XML's role in enterprise applications. Conversion utilities to map XML documents to popular databases, tools for SOAP design, Web service generation utilities, XML Schema designers, XSLT programming environments, and other such features are becoming a part of these IDEs. In the current market the XML IDEs have matured to serve the needs of developers building enterprise applications. And, as was inevitable, the XML vendors are now stepping into the space previously occupied by vendors who offered IDEs for software language platforms such as Java, C, and C++.

Our February issue featured an interesting article by Jim Gabriel on the state of XML development tools in the market. Jim identified several of the requirements for IDEs that support XML development in business applications. It's refreshing to see that tools such as Altova's XML Spy and Tibco's TurboXML have risen to the challenge posed by software developers in a continuously evolving marketplace. Some of these tools already provide a large part of the functionality outlined in Jim's article. For example, XML Spy offers a COM API, a JavaScript environment, support for ASP and JSP Web pages, and a host of other features targeted to the breed of developers that Jim identified.

The market for these IDEs is currently shared by very few vendors. However, there is lateral growth in IDEs offered by the programming language environment, and they are already stepping onto each other's turf. For example, the Java IDE vendors offer editors for XML, and XML IDE vendors will soon offer support for Java environments. In fact, Altova's latest release of XML Spy offers enhanced support for J2EE 1.3, including support for EJB deployment descriptors. This isn't surprising; indeed, it's a natural evolution of the toolset, because, as mentioned earlier, EJB deployment descriptors are written in XML. An example of a Web application server that addresses Web development outside a pure Java environment is Macromedia's ColdFusion app server, which provides rich features for XML processing. Attesting to the maturity of the XML-based IDEs, many of them are featured under Java IDEs in the Readers' Choice Awards hosted by *Java Developer's Journal*, one of our sister magazines.

As the market continues to grow, it will be interesting to see how vendors with a pure XML legacy and vendors with a traditional programming platform legacy compete in the same space. Since the XML IDEs have catered to XML development from the get-go, they have more mature products in that space. It's unlikely that traditional IDEs will be able to catch up in this space. Consequently, we should see another round of integration, mergers, partnerships, and acquisitions. And perhaps a new breed of developer will be defined: the XML developer! ☛

AJIT @ SYS-CON.COM

AUTHOR BIO

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

SilverStream Software, Inc.

www.silverstream.com



WRITTEN BY ISRAEL HILERIO

The Problem with XML-Based Storage

As Java and XML continue as the de facto standard for developing enterprise applications, issues arise in using these technologies. For example, the need to store XML data, and the criteria for selecting the appropriate repository. Here's a real-world example of how applying XML to the wrong problem can lead to the wrong solution.

I inherited a solution last year from a team that needed to develop a solution quickly and didn't have all the requirements outlined for them. They knew some of the basic structures and relationships between data elements, but they were convinced that the initial schema wasn't going to be the final one. While this is a problem faced by most development teams, they were expected to release new functionality on a weekly basis.

The problem involved the development and maintenance of a B2B fulfillment application. The main components of the fulfillment system were customer catalogs and orders. They chose XML as the mechanism for defining customer catalogs and expression orders. Initially this worked out great for catalogs because they were able to extend and add relationships between existing and new elements on a near real-time basis. This also worked out great for order objects because they needed to share data between multiple systems. Integration with other systems was achieved through the combination of Java servlet channels and XML messages. They ended up selecting a company with an innovative XML repository.

The partnership blossomed until we encountered some operational and growth issues. Some dealt with the maturity of the product. As with most nonrelational databases, some concerned the performance of writes as opposed to small reads. Large reads became a problem as there was no caching or cursor mechanism built inside the engine. HTTP, the mandated protocol to query the database and manage data, isn't the most efficient protocol, as many of you know. While it makes sense from a Web-accessibility perspective, access to the database was within the internal hosted network – from servlets, never from the Web.

We discovered a space reclamation problem. Documents marked for deletion were never deleted at all. This meant that document replacement took twice as much space as document creation. To reclaim space we needed to run a defrag utility every few weeks that would search through the system and delete the marked documents. Furthermore, the instability of the database forced us to reindex it every time it was brought down for backups. An additional side effect was that, in order to reduce the number of persistence engines, some RDBMS behavior inside the XML repository was replicated. While some of this behavior was unique to the vendor they chose, other parts were relevant to most nonrelational XML repository solutions.

At the time, we didn't consider RDBMS tools; we believed our schema had to be adaptable and were fearful of the limitations associated with a rigid schema. Also, it wasn't until recently that RDBMS systems have added credible XML support to their environments. At the request of our customer and to alleviate some of our operational issues, we decided to look into porting the system to an RDBMS with XML support.

Let me outline the lessons we learned:

- While many XML repositories are based on object-oriented databases, the maturity level and operational aspects associated with them are not very robust. When looking at an XML repository, keep in mind that many of the other relationships you need to define inside your application are non-XML. Unless you want to maintain two different systems, one for XML documents and one for data relationships, you'll end up creating relational relationships inside the XML repository. Based on our findings, there's little benefit to this approach.
- Identify the speed requirements of your system. If your application is doing a lot of writes and very few reads, it won't benefit from an XML repository.
- Define the data access protocols. Are they binary based like JDBC, or character based like HTTP?
- Evaluate the operational utilities of the solution. What is the procedure to perform a backup? How long does it take to perform a 10GB backup? How are backups restored? Are there any memory leaks or space reclamation problems? How often is the repository required to be indexed?

In summary, don't assume. You know what they say! ☛

IHLERIO @ INNOVATEM.COM

AUTHOR BIO

Israel Hilerio, PhD, is chief software architect for Metavonni, a buyer management software company that develops enterprise solutions that allow corporate suppliers to manage their complex relationships with multiple buyers.

XML's Contribution to Web Services



— [REVIEWED BY BOB SWART] —

AUTHOR BIO

Bob Swart (aka Dr.Bob at www.drbob42.com) is an IT consultant, author, trainer, and Webmaster of Bob Swart Training & Consultancy (eBob42) based in Helmond, The Netherlands. Bob has spoken at Delphi and Borland developer conferences since 1993. Coauthor of several books, including C++ Builder 5 Developer's Guide, Bob is also a contributor to a number of computer magazines.

DRBOB @ CHELLO.NL

Professional XML Web Services

by Patrick Cauldwell, Rajesh Chawla, Vivek Chopra, Gary Damschen, Chris Dix, Tony Hong, Francis Norton, Uche Ogbuji, Glenn Olander, Mark A. Richman, Kristy Saunders, Zoran Zaev

Published by WROX Press

Pages: 803

List Price: \$59.99

Professional XML Web Services isn't really an XML book, but more a pure Web services book. The XML part is because XML is used to represent Web services (in the WSDL, for example). The book covers many types of Web services, not just XML Web services (a term that's like a *binary executable*).

The book consists of 15 chapters. That's not much, if you take into account that it was written by 12 authors. In fact, no author has written more than two chapters. I must admit that my feelings about the result are mixed. Some chapters flow really well and are clear to read, but others take more time to swallow. And although it's usually a good idea to let each author write about his or her favorite topic, in this case it also results in a lot of information that's repeated. The best way to view this book is just as a collection of 15 really long papers. That way you can select a few papers that you need to read at a single time without having to read the book from cover to cover.

The 15 chapters are divided into three parts. The first part contains an introduction into Web services (including a historical overview, which is nice to read) as well as supporting techniques (such as the HTTP and SMTP protocols). While the first chapter may provide interesting background information, there was little news in the second, so I was glad to finish the first part of the book after 75 pages and start on the real content.

The second part covers the Web service languages, in a more or less cross-language way. If this sounds confusing, then allow me to explain that by *Web service language* the authors mean WSDL and UDDI, while the computer languages to implement Web services are covered in the third part of the book (coming up). The second part contains five chapters, about SOAP Basics, SOAP Bindings, WSDL, UDDI, and implementations of UDDI. The first chapter defines the SOAP protocol, explains about the header, body, and envelope, and provides the groundwork for the following chapters. The chapter about SOAP bindings introduces some examples of the toolkits that will be used more extensively in the third part of the book. The last two chapters of this second part of the book were fortunately written by the same author, since they cover UDDI (the specification) and UDDI implementations (using Java as the main language and IBM's UDDI4J as the main implementation tool). At 112 pages (combined) this would be a bit too much for one chapter.

The third part of the book covers Web services implementations, with chapters for

Microsoft SOAP Toolkit 2.0, Easy Soap (C++), SOAP::Lite (Perl), SOAPx4 (PHP), Python, and Apache and Tomcat for Java-based implementations. For most of these implementation tools or environments, a similar set of examples is built, so we can compare them against each other.

It would be nice to have a chapter with an overview about the strong and weak points, but I guess that's one of the disadvantages of having a dozen authors: there's probably no single author with the entire big picture overview to write such a comparison. As a related issue, since a number of tools are covered, we never get into a lot of depth. Or at least not into the level you'd hope for. But perhaps this book will be followed by specific SOAP books, such as the recently published *Professional Java SOAP* (also from WROX Press).

Anyway, after the tools themselves are covered we get some interesting chapters about "advanced" topics such as Web services security, and two case studies (a Java filesystem case study and the WROX auction case study).

The appendices start with a reprint of the Simple Object Access Protocol (SOAP) 1.1, taken from the W3C note issued on May 8, 2000 (in Appendix A), and the Web Services Description Language (WSDL) 1.1, taken from the W3C note issued on March 15, 2001, and finally material from the UDDI version 2.0 Data Structure Reference UDDI Open Draft Specification from UDDI.org, dated June 8, 2001. The index is a bit short at 25 pages, but uses a tiny font, so a lot of keywords are listed (without significant bias toward one chapter or another).

The book focused mainly on SOAP 1.1. Before it was published (in September 2001), the first Working Draft of the SOAP 1.2 Specification was made available, but the introduction (as well as Chapter 3 on SOAP basics) explains that since there were no implementations supporting this Working Draft, they decided not to cover it in any detail. Too bad that the back cover promises "an overview of SOAP 1.2" next to a "detailed explanation of SOAP 1.1." But perhaps a new edition later in 2002 will remedy that.

The book doesn't contain any source code, but as usual the source code samples can be downloaded (2.54MB) from the WROX Web site, where you can also read a sample chapter (Chapter 3 about SOAP basics). Apart from the WROX Web site, you can also visit www.drbob42.com/books for other book reviews.

I enjoyed most of the chapters, and although I don't use all the languages and implementations that were covered, I still think this is a book I'll get back to once in a while. If you're looking for a book to learn about SOAP and Web services using different tools such as Java, C++, Perl, PHP, and Python, this is a good one. If you already know which tool you'll be using, then you may want to look for a specific book for your SOAP environment, to get more in-depth examples and coverage. ☺

BEA

www.bea.com/download

A large satellite dish antenna is the central focus of the image, angled upwards towards the top right. The dish is a complex structure of white metal struts supporting a green mesh surface. It is mounted on a white base with various mechanical components and pipes. The background is a sky filled with soft, white and grey clouds. The title 'XML' is written in a very large, bold, black serif font across the top half of the image. To the left of the title, the text 'PART 1 OF 2' is written vertically in a smaller, white, sans-serif font. In the bottom left corner, there is a small black box containing the page number '10' and the text 'volume3 issue3'. In the bottom right corner, the website 'www.XML-JOURNAL.com' is written in a white, sans-serif font.

XML

PART 1 OF 2

WRITTEN BY Mike Jasnowski

AUTHOR BIO

Mike Jasnowski, a senior software engineer with eXcelon Corporation in Burlington, MA, has over 17 years of experience, including more than 4 with Java. Mike is a Sun-certified Java programmer.

MESSAGING WITH JAXM

How to exchange SOAP messages

I could begin this article by expounding on the many applications that XML has found itself in, but that would be pointless. It's well known that XML has found more than a niche in a variety of applications, in all tiers of applications, and in all types of applications.

Messaging is one area of application development in which XML is being used more and more. Traditionally, messaging has been done using technologies like the Java Message Service, in which applications exchange data by sending the data to a queue; the data is then taken off the queue and processed by the intended recipient.

Applications that exchange messages must be concerned with how the message will be transported to the recipient. Protocols such as HTTP, FTP, and SMTP are used to define a standard transport for packaging information at a higher level.

Applications must also be concerned with how the payload or body of the message is encoded. Even though XML has become a popular encoding for exchanging messages, the need to have a standard encoding still exists. If every entity exchanging messages had its own encoding, it would be a nightmare to try to make sense of every message you received – especially when your application may be concerned only with the payload of the message being sent, not necessarily how it got there.

Standards Are Needed

With the many variables present in application development, standards are evolving that will help speed the development and adoption of XML for messaging. Standards already exist for the various protocols that may be in use during execution of the application, ensuring a common understanding between applications that need to communicate. These standards are also being applied toward a common API for using these different technologies together.

Sun Microsystems, along with a host of other contributors, has developed the Java API for XML Messaging, or JAXM. JAXM provides a mechanism for exchanging SOAP messages. Using SOAP for the envelope to package your messages ensures a standard way to package them. At the time of this writing, Sun had just released the 1.0 version of the specification. The examples in this article were written using an Early Access version of the API.

Part 1 of this series focuses on utilizing JAXM to send SOAP messages directly to a recipient. Other functionality will be mentioned briefly, but I'll save the discussion for Part 2.

SOAP Basics

If you're not familiar with SOAP, I'll give you a brief introduction. If you are, skip ahead to the section titled "JAXM Overview."

SOAP – Simple Object Access Protocol – is an XML-based protocol for exchanging messages and executing remote procedure calls. Developed before JAXM, it continues to be developed as an independent specification by the W3C (www.w3c.org). SOAP specifies a schema that defines a set of elements and attributes that allow you to construct a SOAP message. For RPCs SOAP uses the XML Schema recommendation for representing serialized types. A simple SOAP message might look like the following:

```
<Envelope xmlns=http://schemas.xmlsoap.org/soap/envelope/">  
<Header/>  
<Body>XML Journal</Body>  
</Envelope>
```

SOAP messages define an envelope, headers, and a body to construct the SOAP message. A complete discussion of SOAP is beyond our scope, but an understanding of the general construct of a SOAP message is sufficient to send the messages in this article. The contents of the body of a SOAP message are user-defined, meaning you can pretty much put anything you want between the <Body> start and end tags – XML, for example, or plaintext, or a base64-encoded file that you want to send.

I previously mentioned that SOAP can be used to execute remote procedure calls. While JAXM can send SOAP messages, constructing an RPC for all but the simplest calls is nontrivial due to the serialization procedures that take place. You can learn more about executing these calls using SOAP by visiting <http://xml.apache.org> and looking at the Apache SOAP implementation.

Now that I've gotten some SOAP basics out of the way, let's take a brief look at HTTP. Again, if you're familiar with HTTP, please skip to the "JAXM Overview" section.

HTTP Basics

HTTP provides a standard network transport protocol. It's used when a Web browser talks to a Web server, for example. The browser and server understand the construct of an HTTP message and can respond to each other accordingly. HTTP messages come in two varieties: a request message and a response message. The former might ask for a resource such as an HTML page; the response to that message would be the resource itself or an error response.



HTTP request and HTTP response messages contain a set of headers that provide the recipient of the message with information about the request or response. Headers in an HTTP message are specified using a string indicating the name of the header followed by a colon and then a space. Following the space is the actual value of the header. The specific syntax requirements for HTTP headers can be found following the links at the end of this article.

Along with the headers, an HTTP request has a request URI indicating the resource requested. An HTTP response, on the other hand, has headers, as well as a status code and reason phrase indicating the success or failure of the request, followed by any content the request generated. This content could be the resource requested or an error message indicating failure of the request.

A typical HTTP request message might look like this:

```
GET /page.html HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: localhost:8888
Connection: Keep-Alive
```

Here the request indicates that the resource requested is page.html located in the root directory. It also specifies some information about the requestor using the User-Agent header and that the Host being accessed is the local machine on port 8888.

A response to this message might look like this:

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 65
Last-Modified: Mon, 26 Nov 2001 19:33:00 GMT
Servlet-Engine: Tomcat Web Server/3.2.2 (JSP 1.1; Servlet 2.2; Java 1.3.1;
Windows 2000 5.0 x86; java.vendor=Sun Microsystems Inc.)

<HTML>
<BODY>
  This is the page.html resource
</BODY>
</HTML>
```

Here the combination of a status code of 200 and a reason phrase of OK indicates that a successful HTTP request was made. The Content-Type header indicates that the mime-type of content in the payload is HTML; the Content-Length header indicates that the number of bytes in the payload is 65. Following the last header are two carriage-return line-feed pairs, which is the way the HTTP message indicates that the end of the headers has been reached and content is coming next. The contents of the page.html file are embedded as the payload of the HTTP message.

This simple example of an HTTP request and HTTP response can be applied to sending SOAP messages. The HTTP request shown used the HTTP GET method to request the resource named page.html, and the contents of the page.html file were sent back in the HTTP response.

SOAP messages are sent using the HTTP POST method, so a typical SOAP message might look like this:

```
POST /SOAPProcessor HTTP/1.1
Content-Type: text/xml
Content-Length: 112
Host: localhost:8888
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Header/>
<Body>XML Journal</Body>
</Envelope>
```

In this example the SOAP message is sent to the local machine on port 8888, as specified in the Host header. The URI of the request is /SOAPProcessor, which could be mapped to a JavaServer Page or a Java servlet that will process the SOAP message. The Content-Type header

indicates that the payload of the HTTP message is mime-type text/xml and the length of the data is 112 bytes.

SOAP and HTTP together provide complementary protocols for exchanging information. HTTP is used as the network transport protocol and SOAP is used as the message packaging protocol. Let's look now at how JAXM makes use of these two technologies and keeps the developer from having to know how to construct an HTTP message or a SOAP message.

JAXM Overview

JAXM provides the necessary framework for exchanging messages in two basic scenarios. The first, *synchronous messaging*, involves sending a SOAP message directly to another recipient and waiting for the response. The second scenario, *asynchronous messaging*, involves sending a message to another recipient via a messaging provider. The response to the message will come later. This is conceptually similar to the way one application can send a message to another application using JMS. Figure 1 illustrates how JAXM operates and fits into the overall picture of an application.

The JAXM API has the classes and interfaces for creating and packaging SOAP messages as well as for sending and receiving them. The contents of the SOAP message must be constructed using other means, that is, whatever goes between the <Body> tags is user-defined. JAXM provides support for constructing SOAP messages as well as SOAP messages with attachments. This could be used to send a purchase order along with a picture of the item being purchased.

JAXM defines the connection to a recipient of a message as an *endpoint*. When you want to send a message, you create an endpoint using the address of the recipient. I'll show you later how to create an endpoint for sending a message directly to a recipient. In Part 2 you'll see how the endpoint of the recipient is specified in a profile.

The JAXM classes and interfaces are in two packages: javax.xml.soap and javax.xml.messaging. The former is used to construct SOAP messages and to send messages without a messaging provider. The latter contains classes and interfaces for working with messaging providers, as well as some utility classes for creating endpoints.

Earlier I spoke of how protocols like HTTP and FTP come into play when exchanging messages. JAXM currently supports sending SOAP messages via HTTP. Now that we have a bit of the overview out of the way, let's take a closer look at the classes and interfaces that make up the implementation. I won't cover every interface and class, just the primary ones.

Creating SOAP Messages

To create SOAP messages with JAXM, you must first get an instance of a class called the MessageFactory, which can create instances of the type of message desired. There are two ways to create a MessageFactory instance. The first is for use when sending messages without a messaging provider; the second is for use when using a messaging provider. Here I'll cover the former; Part 2 will cover the second way.

Instances of the MessageFactory can be created using the newInstance() method of the MessageFactory class as follows:

```
MessageFactory messageFactory = MessageFactory.newInstance();
```

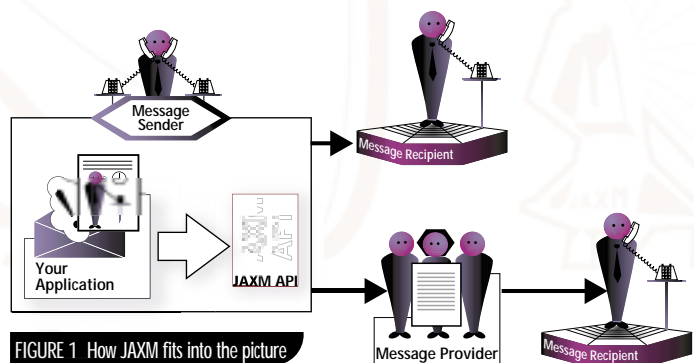


FIGURE 1 How JAXM fits into the picture

HitSoftware

www.hitsw.com



Once you have the `MessageFactory` instance, you can create an instance of a `SOAPMessage` class using the `createMessage()` method.

```
SOAPMessage soapMessage = messageFactory.createMessage();
```

The `SOAPMessage` instance you have after the call to `createMessage()` results in an empty SOAP message. You need to populate it with the information you want to send. To do this I need to get at the body of the SOAP message, which I can't access directly using the supplied classes. I first need to access the envelope – then I can access the body of the SOAP message.

To access the envelope I use the `getSOAPPart()` method to get an instance of the `SOAPEnvelope` interface. Having gotten the instance, I can use the `getBody()` method to retrieve an instance of the `SOAPBody` interface. The following lines of code illustrate how to add the text “XML-Journal” to the body of a SOAP message:

```
SOAPPart soapPart = soapMessage.getSOAPPart();
SOAPEnvelope soapEnvelope = soapPart.getEnvelope(true);

SOAPBody soapBody = soapEnvelope.getBody();
soapBody.addTextNode("XML-Journal");
```

The second line of code deserves some explanation. The `getEnvelope()` method retrieves the current envelope for the SOAP message. This method takes a Boolean indicating whether a new <Header> element should be created and added to the message.

Examining the SOAP Message

The `SOAPMessage` class includes a method called `writeTo()` that allows you to dump the contents of a SOAP message. This ability is useful if you're debugging a SOAP message, but has no practical use during normal execution of a JAXM client. The following code snippet shows how to use the `writeTo()` method to send the SOAP message to `System.out`.

```
/* Create an empty SOAP Message */
SOAPMessage soapMessage = messageFactory.createMessage();

/* --- Add SOAP message content here ----*/

soapMessage.writeTo(System.out);
```

If I were to dump the previous SOAP message, it would look like the following:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Header/>
<Body>XML-Journal</Body>
</Envelope>
```

The existence of the <Header> element is determined by passing a Boolean value of true to the `getEnvelope()` method, as shown previously.

Adding an Attachment

Adding an attachment to a SOAP message means attaching the contents of an external resource such as a text file or binary image file as part of the message. This is the same concept as adding attachments to an e-mail message. In fact, the JAXM API makes use of the JavaMail API as well as the JavaBeans Activation Framework for working with attachments in JAXM. The following code snippet shows how you can add an attachment to a SOAP message:

```
/* SOAP message created prior to executing these lines of code */
URL url = new URL("http://localhost:8080/tomcat.gif");
AttachmentPart part1 = soapMessage.createAttachmentPart(
    new DataHandler(url));
```

```
soapMessage.addAttachmentPart(part1);
```

Adding attachments is entirely optional – they just add a level of functionality. For instance, instead of encoding a binary image file and adding it to the body of a SOAP message, you can use the attachments feature to do this. The ability to add attachments is mirrored by the ability to extract attachments in a SOAP message.

Sending a SOAP Message

To send a SOAP message you first need to establish a connection between the sender and the receiver. Here I'll focus on sending messages without a provider, which involves creating an instance of the `SOAPConnection` class:

```
SOAPConnection soapConnection = SOAPConnection.newInstance();
```

After creating the connection, you can use the `call()` method to send your SOAP message. This method takes two arguments: (1) the instance of the `SOAPMessage` class, and (2) an instance of the `URLEndpoint` class. Earlier I spoke of how JAXM uses endpoints to define the network endpoint to send a message. The JAXM API defines the `Endpoint` class and a subclass named `URLEndpoint` that's used to define an endpoint for sending messages directly to a recipient. Listing 1 shows the source for an application that sends a SOAP message using the JAXM API.

Before compiling and running the example, make sure you have the proper libraries in your CLASSPATH.

In the JAXM distribution is a directory called `lib` that contains the additional libraries and a JAR called “`jaxm.jar`” that contains the classes and interfaces but not the implementations of the interfaces. Therefore, it's important that you also be aware of the directory “`jaxm`” that contains the `client.jar`, which does contain the implementations, including the default `MessageFactory` implementation.

Receiving a SOAP Message

Before I can execute the JAXMClient you just saw, I need to create the receiver of the SOAP message. It doesn't have to be a client that uses the JAXM API. In some cases, though, for convenience, you may want to make them both use the JAXM API because it has the necessary classes to work with SOAP messages.

In the case of a SOAP message sent directly to a recipient, the receiver can be anything that can process an HTTP POST and also understand SOAP messages. The easiest way to set up a SOAP receiver is to create a JSP that sends a synchronous response to the message. For this article I used Tomcat 3.2 and wrote the following JSP code:

```
<%@ page contentType="text/xml" %>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
Message Received!
</Body>
</Envelope>
```

This JSP responds with the text “Message Received!” in the body of the SOAP message sent back to the sender. The JSP doesn't do anything with the SOAP message sent to it, of course; it simply acknowledges that it was received. To process the message and return a more meaningful response, we'd have to examine the contents of the HTTP message and parse it into a SOAP message.

Processing the SOAP Message

To process the message I extract the payload of the HTTP message and parse it into a DOM Tree. I'll append the contents of the SOAP message that was sent to another string that I'll send back. In Part 2 I'll show you how to use the JAXM JSP tag library to process the SOAP message. In this example I've used the Java API for XML Processing (JAXP) to extract and create a DOM Tree representing the SOAP message. I then

get the node that represents the actual content of the body of the SOAP message using customary DOM Node interface methods.

Listing 2 shows the source for extracting the SOAP message and the body of the SOAP message. This is a slightly updated version of the JSP you saw in the section “Receiving a SOAP Message.”

The result of running this JSP would result in a SOAP message that looks like this:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
Message Received! You sent: XML-Journal
</Body>
</Envelope>
```

I’ve inserted carriage returns and line feeds after each line of the XML to make it more readable for the purposes of this article. The actual response would be an XML document all on a single line. But as you can see, the content of the SOAP message “XML-Journal” was appended to the body of the SOAP message.

Processing the Response

When the response is sent back, the call() method I used earlier returns an instance of a SOAPMessage object. I can extract the response using the methods of the SOAPMessage class. You can extract the content of the <Body> element using a code snippet such as:

```
SOAPMessage soapReply = soapConnection.call(soapMessage,
new
URLConnection("http://localhost:8080/processsoap.jsp"));
SOAPPart soapReply = reply.getSOAPPart();
SOAPEnvelope replyEnvelope = soapReply.getEnvelope();
SOAPBody replyBody = replyEnvelope.getBody();
System.out.println(replyBody.getValue());
```

The last line of code will print the line “Message Received!” that’s sent back from the JSP, plus the value appended by the JSP in Listing 2. One thing about using the SOAPMessage class is that it’s somewhat DOM-like in that you must access parent elements (e.g., the envelope) before you can access the body of the message.

Summary

In this article you’ve seen how messages can be exchanged using SOAP and HTTP, and how JAXM uses these two technologies to exchange SOAP messages. The JAXM API has the necessary classes for working with SOAP messages as well as sending them and receiving the response as a SOAP message.

• • •

In Part 2 I’ll discuss sending SOAP messages using a messaging provider. You’ll see how to specify a messaging provider and write a JAXM client to send messages to it, and how to write a JAXM client that receives messages asynchronously.

You’ll also examine the JAXM JSP Tag library that enables you to send and receive SOAP messages using JSP custom tags.

Links

The following links will provide you with more information about SOAP, XML, and messaging in general, as well as locations for downloading the APIs discussed in this article.

- *Java API for XML Messaging*: <http://java.sun.com/xml/jaxm>
- *Simple Object Access Protocol 1.1 (SOAP w/Attachments)*: www.w3.org/TR/SOAP-Attachments
- *Simple Object Access Protocol 1.1 (SOAP)*: www.w3.org/TR/SOAP
- *Java Message Service*: <http://java.sun.com/products/jms/index.html>
- *Hypertext Transfer Protocol 1.1*: www.w3.org/Protocols/rfc2616/rfc2616.html 

LISTING 1 JAXMClient.java (MJT100012.txt)

```
import javax.xml.soap.*;
import javax.xml.messaging.*;
import java.io.*;
import java.net.*;
import javax.activation.*;

public class JAXMClient{

    public static void main(String args[])
        throws SOAPException,IOException{

        /* Create the MessageFactory */
        MessageFactory messageFactory =
            MessageFactory.newInstance();

        /* Create an empty SOAP Message */
        SOAPMessage soapMessage =
            messageFactory.createMessage();

        SOAPPart soapPart =
            soapMessage.getSOAPPart();
        SOAPEnvelope soapEnvelope =
            soapPart.getEnvelope(true);

        SOAPBody soapBody = soapEnvelope.getBody();
        soapBody.addTextNode("XML-Journal");

        SOAPConnection soapConnection =
            SOAPConnection.newInstance();
        SOAPMessage replyMsg =
            soapConnection.call(soapMessage,
                new
                URLEndpoint("http://localhost:8080/test/processsoap.jsp"));
        soapConnection.close();

    }
}
```

LISTING 2 processsoap.jsp (MJT200012.txt)

```
<%@ page contentType="text/xml" %>
<%@ page import="java.io.*" %>
<%@ page import="org.w3c.dom.*" %>
<%@ page import="org.xml.sax.*" %>
<%@ page import="javax.xml.parsers.*" %>
<%

Document doc = null;

DocumentBuilderFactory dbf =
    DocumentBuilderFactory.newInstance();

DocumentBuilder db = null;
try {
    db = dbf.newDocumentBuilder();
} catch (ParserConfigurationException pce) {
    System.err.println(pce);
}

try {
    doc = db.parse(request.getInputStream());
} catch (SAXException se) {
    System.err.println(se.getMessage());
} catch (IOException ioe) {
    System.err.println(ioe);
}

String result = "";
Node envelopeNode = doc.getDocumentElement();
Node bodyNode = envelopeNode.getChildNodes().item(1);
Node textNode = bodyNode.getFirstChild();
result = textNode.getNodeValue();

%>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
Message Received! You sent: <%= result %>
</Body>
</Envelope>
```

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



Framing enterprise messaging

The World of Enterprise Messaging

Messaging has been a part of computing from the minute that computers were enabled to persist data across invocations of process. It was inescapable that before long machines would be leaving messages for other computers, soon to be followed by people leaving electronic messages for other people.

Messages are a critical component of enterprise computing and they come in all sizes and volumes. However, ask 10 IT specialists to define enterprise messaging and you're likely to get 10 different answers. This variance is due to the overloaded meanings assigned to *enterprise*, *messaging*, and *enterprise messaging*. Moreover, how these terms are used expands daily through the introduction of new technologies.

Based on my opening paragraph, giving a definition of enterprise messaging here would be arrogant, as the definition provided would merely be the opinion of the author. Instead, I'll provide a view into this messaging space and discuss the uses and derivations of the messaging paradigm as it's used in the enterprise setting.

Here are some simple examples:

- **Enterprise Application Integration:** Applications sending messages to each other to query, update, and aggregate information
- **Web services:** Companies exposing internal business processes via a well-defined XML interface operating over HTTP or SMTP
- **Electronic Data Interchange (EDI):** Document exchange for the purposes of data exchange and transaction processing
- **Alerting/notification:** Applications telling users that a particular event has occurred; events can be disseminated over multiple messaging technologies
- **Live data feeds:** Large volumes of continuously streamed data messages used to provide real-time or continuous processing

These examples illustrate the business case for messaging; they also demonstrate the diversity of messaging technologies used in the enterprise.

Messaging Protocols

Scoping the messaging market is complex and extensive. However, it's an integral part of understanding how expansive this one area of the industry is. What follows is a list of the messaging protocols most often found in enterprise environments. Associated with each is a description of the technology and how it's most often used. (Those experienced with messaging systems may find this section redundant, but it's an important foundation for understanding the remainder of the article.)

Hypertext Transfer Protocol

The most popular messaging facility in the world. Often overlooked as a component of the messaging industry, HTTP is a request/response protocol that delivers content enveloped with specific headers – a common attribute of many messaging systems.

Simple Mail Transfer Protocol (SMTP/E-mail)

The most common form of messaging found in the enterprise today. The number of such messages delivered daily measures in the millions. It is the most popular form of electronic communication between people within and across enterprises. E-mail is most often described as a store-and-forward paradigm, where messages are stored until the recipient reads them.

Instant Messaging

Quickly closing in on e-mail as the heaviest and most popular method of electronic communication between people. In contrast to e-mail, instant messaging messages are intended for immediate delivery versus being stored until the recipient checks. One of the biggest hurdles to instant messaging in the enterprise is the lack of standards and security.

Paging

After instant messaging, the most popular form of electronic messaging. Usually initiated over a phone connection or via a Web page, pages are delivered to wireless devices capable of handling small alphanumeric or numeric messages.

Short Message Service

A protocol that facilitates the delivery of short alphanumeric messages to pagers and wireless phones. What's important about SMS is that it's a store-and-forward facility that ensures delivery of the message once the device comes into range to receive. Additionally, it's an important mechanism to send notifications when users can't use voice equivalents.

Interprocess Communications

In most enterprise multitasking operating systems, processes run in separate and protected memory regions, which ensure that one process can't corrupt the data running in another process. Sometimes, however, it's necessary for multiple processes to pass data to each other and to synchronize access to resources. A form of messaging, IPC is used to overcome the barrier to cross-process communication and allow processes to communicate on a single machine.

Remote Procedure Call

Perhaps the most complex of all mes-

AUTHOR BIO

JP Morgenthal, CTO of Ikimbo, is an internationally prominent expert on the design and implementation of distributed systems for the enterprise and underlying technologies: Java, XML, EAI, and B2B.

VoiceXML Planet Conference & Expo

<http://events.internet.com>

.Net Developer Conference & Expo

<http://events.internet.com>

saging technologies, though this may sound simplistic. RPCs allow an application to execute functionality on a remote system as if it were executed locally. In many ways the RPC is an enhanced IPC mechanism that operates across geographical boundaries. When a system executes a function locally, its symbols and registers are available on the same machine as the calling machine. However, with RPCs the function is executed remotely and that function's state is isolated on a remote machine. This means the RPC mechanism is responsible for carrying the parameters to the executing machine, executing the function on the remote machine, and either returning the function value or telling the calling machine that there was a failure.

In addition to these protocols, which are mostly responsible for enabling the messages to be handled at an application level, there are messaging infrastructures on which these protocols ride. Examples are 802.3 (Ethernet), TCP/IP, UDP, 802.11b, Mobitex, Motient, GPRS, and CDMA. There's no need to expand on these groups at this time; however, it's important to realize how they further expand the messaging technology space.

Messaging Categories

The previous section described many different protocols used in enterprise messaging technologies. From that list you can see that the industry is wide and covers such a diverse set of attributes that the only way to categorize messaging based on that list is to say that data moves from a sender to a receiver. However, the aforementioned technologies can be grouped by other features they share. Some examples follow.

Asynchronous Messaging

In asynchronous messaging the sender doesn't block – stop processing – until it receives a response. That is, once the sender delivers the message to either the delivery agent or the recipient, it's done with its task. E-mail, instant messaging, IPC, paging, and SMS are all examples of asynchronous messages.

Synchronous Messaging

In contrast, synchronous messaging stops all processing until it receives a response or reaches a threshold of waiting. HTTP and RPC are examples of synchronous messaging.

Queueing

An underlying infrastructure for communication across disparate enterprise systems and applications, this form of messaging creates a network of virtually

named queues that applications use to communicate with each other – much as people communicate with each other over e-mail. Typically, one application sends information to another application, which may or may not be active, by placing a message in its queue. When the receiving application is activated, it processes all the messages waiting for it in a set of named queues. This technology is typically used to enable communications between two specific systems. E-mail, SMS, and paging are examples of queue-based messaging.

Publish/Subscribe

Often called pub/sub, this technology allows clients to subscribe to a particular content delivery mechanism. When the source wants to distribute content, it first delivers it to the pub/sub system, which is then responsible for delivering it to all the subscribers. This method can be very effective for delivery of a single piece of information to a large community of users. Instant messaging is a form of publish/subscribe messaging, especially when used for multiperson conferences.

Point-to-Point

Another way to describe messaging technologies is by the topology that they operate over. Some messaging protocols only operate directly. This means that the sender will deliver messages only to the recipient without going through a delivery agent. IPC, HTTP, and RPC are examples of point-to-point messaging technologies.

Hub-n-Spoke

This is another messaging topology in which a spoke (node) delivers its message to the hub, which is responsible for delivery of the message. Electronic mail is an example of hub-n-spoke.

Guaranteed Delivery

In guaranteed delivery messaging environments there is a contract between the sender and the delivery agent that every attempt will be made to deliver the message or a negative delivery response will be generated. All of the above messaging technologies can operate in a lossy or guaranteed mode.

Enterprise Messaging Integration

As indicated in the earlier sections, the enterprise relies daily on a large number of messaging technologies. Some are managed directly by the enterprise, while service and telecom providers manage others. This is where, one might say, "the rubber meets the road." The enterprise is one of the most demanding user communities – once reliant on a service, that service is expected to run with a certain

degree of reliability and performance ...and sometimes security.

Now let's look at a real-world scenario where this infrastructure could have significant impact on the enterprise if these systems aren't well understood and highly available. In the newly emerging ebXML messaging model, SOAP messages are sent from one company to another for the purposes of querying a repository, facilitating an electronic transaction, or initiating an external communication. ebXML messages are designed to operate over a variety of messaging technologies, but most notably SMTP (e-mail) and HTTP (Web).

SOAP is essentially a packaging facility designed to support wrapping of data for delivery over multiple messaging technologies. In addition, one of the more popular information sets often packaged inside SOAP messages is used to execute RPCs. Hence the message will be received by an HTTP server, which passes it to a SOAP processing engine (if e-mail is used, a polling thread will pull the message off the server and pass it to the SOAP processing engine). Next, the SOAP processing engine will send the document to the proper server process based on the information provided in the associated WSDL (Web Service Definition Language) document.

This example illustrates how a single business process touches multiple messaging technologies. At each point, the individual messaging technology may encounter exceptions or may not be available. The flow of the data across these different messaging systems characterizes the term *enterprise messaging* and its requirements.

In today's enterprise messaging world, the messaging infrastructure is a key indicator of the IT capabilities of the organization. Strong IT environments require high availability and redundancy for key messaging services, such as Web and e-mail, and mission-critical application messaging services, such as queues. Moderate and weak IT organizations will attempt to support these services on a shoestring, resulting in downtime and lost messages. Or they may host these services, which produces a loss of control over connectivity or a need to deal with spotty technical support.

Given the considerations of our earlier real-world scenario, the important factor is that the enterprise has enough scale, reliability, security, and management controls to ensure that the messaging mechanisms employed do not hinder business, but rather support the reliance that the enterprise places on them once they're integrated into the organizational processes. ☛

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


XML GUIDE

AN XML WORKFLOW AND INTEGRATION LAYER FOR TELECOMMUNICATION PROVIDERS

AUTHOR BIO

Ron Ben-Natan, chief technology officer at ViryaNet Inc., previously worked for Intel, AT&T Bell Laboratories, Merrill Lynch, and J.P. Morgan. He holds a PhD in computer science in the field of distributed computing and has been architecting and developing distributed applications for almost 20 years. The author of CORBA, Objects on the Web, and CORBA on the Web, Ron coauthored IBM San Francisco Developer's Guide and IBM WebSphere Starter Kit.

Written by Ron Ben-Natan



There are a number of skeletons in the closets of today's telecommunication service providers. One of the scariest is that most service providers cannot successfully deliver on the promises they make as service-level commitments to their customers. Not *do not* or *will not*. *Cannot*. For many reasons, successful service-level management in the competitive telecom arena remains a theory – full of unfulfilled potential and many broken promises.

The OSS

When I was younger I had a sign on my monitor that read "to err is human; to really make a mess requires a computer." This is a bit misleading since the problem isn't with computer systems per se. It results from a correlation between complex businesses and businesses that utterly depend on software.

The telecommunication industry is one of the most complex industry verticals in the world. Telecommunication providers are faced with many technologies (SS7, ATM, IP telephony, PBXs, etc.) that need to operate at very high efficiency and very low margins. They serve a huge consumer base and offer very diverse products. It's safe to say that as a business operation, telecommunication providers are among the most advanced in the world.

In order to manage such a complex operation, these providers have many IT systems that work together to run a smooth operation (at least that's the intent). These systems can be part of the customer-facing layer or the operations layer. The focus of running an efficient operation is the set of systems categorized as Operational Support Systems – the OSS.

The key to a practical solution to the problems mentioned above is to collaboratively integrate the OSS and develop automated business rules and workflow processes. Especially important is the need to define the workflow process from end to end and to identify and consider all the handoffs and touch points between other organizations, systems, and workflows and other business processes. Every party involved in the workflow must continually collaborate with the others so as to execute effectively. Experience has

shown that most departments or organizations do a credible job within their particular sphere of influence. It's when the order or ticket is handed off (or what one party thought was a handoff) to another organization that the potential for delay, nonperformance, miscommunication, or simply losing track of the job is at its highest.

An Example Nightmare

A real-world example: many large service providers planned for installations of DSL service to be deliverable with a single truck roll when the last mile was leased from the local phone company and the service provider had contracted with a third-party company to do the installations. Sounded good until the technician got on-site and found that the modem was nowhere to be found. Why not? No one had told the customer it was coming and he refused delivery or sent it back. After all, it didn't have his name on it, just the address.

Learning quickly, the technician always kept spares in the truck and usually managed to get the hardware installed. Unfortunately, the technician qualified to do the hardware install wasn't qualified to do the software. So a second technician was sent out, scheduled to arrive a few hours after the first had left.

The end result was indeed chaos. Provisioning times dragged on for months. Actual truck rolls per install exceeded four times the assumption. Worse yet, the company actually failed to provide a process for cancellation of service that included sending someone to retrieve the hardware. Thousands of modems were stranded on customers' premises. Thousands more were lost as part of the drop-shipping mix-ups. For quite a while the service provider in question lost or otherwise couldn't account for over 35% of the modems they had ordered.

The good news is that some of the modems were eventually located – the bad news is that they found them on eBay. (By the way, I still have a modem, having canceled my DSL service a year ago. I'll sell it real cheap if someone wants it.)

A new generation of OSS solutions is providing some light at the end of the tunnel, based on two central concepts: integration and automation. In essence, what providers need is a glue layer on which the systems can reside, a glue layer that is strong enough to drive all of the provider's operations, yet flexible enough to adapt to changing markets and offerings. This layer is based on a merging of integration techniques and automation techniques. Together, the two form a layer that all systems of the OSS plug into. These systems include provisioning systems, trouble ticketing systems, workforce management systems, asset management

Some concepts in this article can be attributed to work the author has done for the upcoming book *Integrating Service Level Agreements: Optimizing Your OSS for SLA Delivery* by John Lee and Ron Ben-Natan, copyright 2002, John Wiley & Sons, Inc.

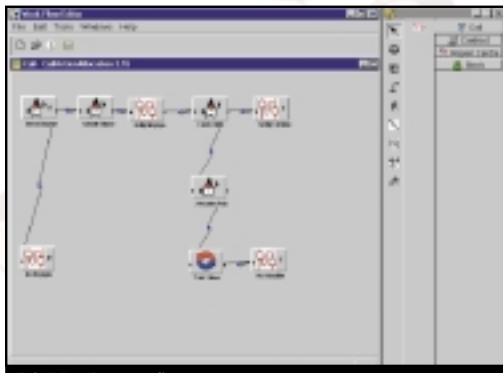


FIGURE 1 Process editor

Parameter	Value
customer	1
process	2
serviceId	3
storeId	4

FIGURE 2 Parameterizing the URL with data in the workflow token

Query	Result
SELECT * FROM
SELECT * FROM
SELECT * FROM
SELECT * FROM
SELECT * FROM
SELECT * FROM

FIGURE 3 Extracting data from the HTTP response using XQL

Style	Result
...	...
...	...
...	...
...	...
...	...
...	...

FIGURE 4 Applying XSL stylesheet to XML description of business process

systems, parts management systems, and customer-facing systems such as billing and customer-care systems.

This glue layer is all about integrating disparate systems into a single virtual operation driven by modeling the business process. It must deal with the merging of different business models into a single business flow as well as with different systems, different platforms, and different data. Because of these circumstances, this new glue is all based on XML.

Business Process Automation

Business process automation is achieved through the use of a workflow management system. Workflow management, simply put, is a technology that allows the automation of processes involving combinations of human and machine-based activities.

Workflow has become a dominant theme in the OSS. Every vendor in the OSS claims to have workflow technology. Unfortunately, *workflow* is a term often misused and sometimes not fully understood. Workflow automates processes in which information and tasks are passed between participants according to a defined set of business rules.

Automation is the key word here: the essence of workflow is the replacement of manual business process management with an automated system – called a workflow management system – that has to support three categories of features:

1. **Build:** The system must provide tools with which a business analyst can design and assemble a process definition comprising the activities that need to be performed, the participating roles, and the flow between the activities based on a set of business rules. An example of a process editor is shown in Figure 1.
2. **Process engine:** The system must include a runtime engine that manages the processes and walks each one through the set of activities as specified by the process definition. The engine needs to perform the actions and evaluate the business rules to determine how the flow proceeds.
3. **Runtime user interface:** The system must also include user interfaces so people can interact with it. The workflows modeling the business processes almost always include a large set of activities that require human intervention; the system must therefore manage a set of screens to allow this. In addition, the process engine should have a user interface component to allow users to monitor and control the engine and the processes running within the engine.

Back-End Integration

So much has already been said about the use of XML to integrate systems. One merely has to look at the various middleware and EAI vendors to see that XML is the preferred data format for integrating between applications, and

that XML-based protocols such as SOAP and WSDL are the industry's answer to decoupled application integration. The key is to look at the data structures defined by the API's signature as an XML document and to use XML transformations (whether XSLT or some other proprietary technology) to do the semantic mapping.

Such API invocations or database accesses occur within the context of a process. It's typical for a query to be performed in one application. Based on the result, an API call is made on one of two separate systems. As an example, a good service delivery process may query a parts-tracking system to see if a modem exists in the technician's inventory. If it does, it invokes a service to reserve that part. If not, it creates a shipment record in a third system so that when the technician arrives on-site the part will be there. It also needs to add a note to the call center agent to inform the customer that a drop shipment should arrive in the next few days. Each call involves taking data from the process, creating an XML document based on the application's API, and invoking the API as a step within the process definition.

Front-End Integration

A workflow engine can also be used for another type of integration or "glue." Integration normally implies database-level or API-based integration. In any case it's a "gluing" of the back ends.

There is, however, another form of integration, one based on the front end. It involves user interfaces that extract and enter data and, in effect, script existing applications as a way to create transactions and integrate systems. This form of scripting has become an important scraping model, especially in Web environments, because it is relatively easy to do with the HTTP protocol.

A Web application is deployed on application servers that get requests through a Web server. Requests come over an HTTP connection and follow a request-response paradigm. Each request has a target, which is the URL, as well as a set of arguments within the HTTP request, either on the URL (if the request is of type GET) or within the body of the request (if the request is of type POST). These arguments are normally used for performing the application functionality. For example, they may be used as the parameters of a query that needs to be performed.

Let's see what a user session in this model looks like. Using a Web browser, a user is working with a Web page. The page displays business information as well as forms into which the user enters information, then presses the button to submit. The browser packages the data entered in the form into the HTTP request, and this is delivered to the server. The server uses this information to perform the transactions or queries (or both).

The server creates the Web page, which displays the result of the queries and the transac-

Now in More

than 5,000

bookstores

Worldwide

Subscribe Now!

FOR FAST
DELIVERY

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

Helping
you enable
intercompany
collaboration
on a global scale

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

**Go
Online
and
Subscribe
Today!**

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**

tions. This page is then delivered back to the browser for the user.

This cycle usually involves a user. Submits are triggered by clicks on a button in a form, and information is viewed and used for making business decisions. But submits really produce HTTP requests, and the data used for making business decisions can be “scraped” from Web pages. In fact, it’s easy to script such interaction with the server programmatically. You can think of this as the modern evolution of screen-scraping software, only this time we’re scraping Web pages and not green screens.

Scripting a Web application requires us to create HTTP requests as well as make use of HTTP responses. We need to be able to create an HTTP request with arguments that are the data elements underlying the queries and transactions. We then must be able to analyze HTTP responses and extract data from them. Finally, we must be able to chain together such cycles in steps and (for example) make use of information we extract from one HTTP response to create input used in a later HTTP request.

A workflow engine is precisely the infrastructure required to perform generic Web application automation and scripting. Specifically, the token (the common data structure throughout the process life cycle) is used as the context through which data structures are passed from one scripting step to the next. Each step in the process defines the URL that will be the basis for the HTTP request as well as the parameters that the request will include. The data elements forming the arguments need to come from the token. An example of such a step definition is shown in Figure 2.

Once the HTTP request has come back, we need to extract information from the resulting page for use later in the process flow. We can use this data in transition rules (to determine what the next application step should be) or as data for use in other HTTP requests. In any case we put the information extracted from the response into the token. Here too we make use of XML; HTML is a subset of XML.

Unfortunately, not all HTML is well-structured XML. Luckily, we have some utilities that will clean up HTML and generate well-formed XML from the returning page. We then use XQL – the XML query language – to pull information from the generated XML. From a definition perspective, what we have to do is define XQL expressions that define which information to pull out of the XML (which is really the resulting page). We then define which token entry each such extraction is injected into, as shown by Figure 3.

Exporting a Business Process Definition

Another use of XML in this framework is the description of the process itself. This process is the core of the OSS as an operation and needs to be communicated throughout organizations. Business processes need to be exported and reused between different parts of the organizations; they need to be easily described as reports and to be read by people as well as by systems. This accessibility to both carbon and silicon is precisely what XML documents are best suited for.

Listing 1 shows an example XML file that represents the steps, transition rules, and attributes of a process. The actual format isn’t important. The part that’s important is the ability to export a process template definition and later import it back into the same or other workflow engine. The example shows the export to be an XML file in the Commerce One SOX format.

While XML is readable by humans, the file shown in Figure 3 isn’t that easy to read. By adding an XSL stylesheet we can go a step further. We can provide an intuitive report format for business analysts who need to view and comment on the process definition. The output from applying such an XSL sheet to the XML document is shown in Figure 4.

Summary

Automation and integration – these are the keys to an efficient operation within many complex businesses that are heavily dependent on computerized systems. Telecommunication providers are perhaps a premier example of this. Within a provider’s world, the OSS is the perfect environment in which a single glue layer supporting integration within an automation platform (all based on XML) can do wonders. ☛

LISTING 1

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet
  type="text/xsl"
  href="http://.../WfExport.xsl"?>
<WfProcessTemplate Identifier="a" IsRef="Embedded">
  <category>
    <![CDATA[Call]]>
  </category>
  <description>
    <![CDATA[Default]]>
  </description>
  <name>
    <![CDATA[CallCompleted]]>
  </name>
  <oid>
    <![CDATA[580000000523]]>
  </oid>
  <origUserMaintenanceFlag>
    <![CDATA[false]]>
  </origUserMaintenanceFlag>
  <steps>
    <WfStepTemplate Identifier="b">
      <apiName>
        <![CDATA[SendEmail]]>
      </apiName>
      <apiProvider>
        <![CDATA[ApiServer]]>
      </apiProvider>
      <apiVersion>
        <![CDATA[1.00]]>
      </apiVersion>
      <defaultPriority>
        <![CDATA[0]]>
      </defaultPriority>
      <description>
        <![CDATA[Commend employee]]>
      </description>
      <immediateActivation>
        <![CDATA[false]]>
      </immediateActivation>
      <inputOutputParameters>
        <WfStepParameterTemplate>
          <additionalKey>
            <![CDATA[ ]]>
          </additionalKey>
          <inOutFlag>
            <![CDATA[false]]>
          </inOutFlag>
          <isApisAdditionalParam>
            <![CDATA[false]]>
          </isApisAdditionalParam>
          <isApisComponent>
            <![CDATA[false]]>
          </isApisComponent>
          <isMapEntry>
            <![CDATA[false]]>
          </isMapEntry>
          <isURLTrackingParam>
            <![CDATA[false]]>
          </isURLTrackingParam>
          <isWAParam>
            <![CDATA[false]]>
          </isWAParam>
          <name>
            <![CDATA[country]]>
          </name>
          <oid>
            <![CDATA[2680000001026]]>
          </oid>
          <paramAdapters>
            <Adapter Identifier="e">
              <sourceEntryName>
                <![CDATA[country]]>
              </sourceEntryName>
              <sourceName>
                <Call/CallComple/1.0/GreatService>
              </sourceName>
              <targetEntryName>
                <![CDATA[country]]>
              </targetEntryName>
              <targetName>
                <![CDATA[ApisInvoker]]>
              </targetName>
            </Adapter>
            ...
          </paramAdapters>
        </WfStepParameterTemplate>
        ...
      </inputOutputParameters>
    </WfStepTemplate>
    ...
  </steps>
</WfProcessTemplate>
```

THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE FREE E-Newsletters SIGN UP TODAY!

Go to www.SYS-CON.com

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

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



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

at www.sys-con.com

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



*Automated markup techniques*

Trends in High Volume XML Publishing

Integrating efficient XML publishing into high-volume content environments remains a significant challenge. Among the many real-world barriers: the need to convert quantities of paper and other legacy documents and to integrate easy-to-use XML publishing tools into the content-creation process, and the lack of workflow management tools necessary for mass conversion environments.

In many environments content creators resist using XML authoring tools, preferring traditional word processing or desktop publishing applications, and simplified “template-style” DTDs are used to accommodate productivity requirements. Consequently, high-volume XML conversions are typically accomplished through “brute force” solutions, where mass OCR (optical character recognition) scanning and tagging are done through expensive outsourcing, often to developing countries where labor for repetitive high-volume publishing tasks is plentiful and inexpensive.

To realize the full benefits of XML for both highly structured and mixed-structure content in high volumes – without the cost, cycle time requirements, and other outcomes inherent in outsourcing – an XML publishing system that minimizes the ongoing intervention of XML programmers is essential. To efficiently convert documents in Word, HTML, PDF, RTE, or other formats into XML, intelligent, rule-based automated markup solutions are required.

For large-volume projects, efficient XML publishing also requires batch processing and workflow management solutions that optimize productivity. In this article I’ll discuss the process requirements for high-volume XML creation and introduce new tools and technologies expressly developed for these mass-conversion environments.

The High-Volume XML Publishing Challenge

Any document can be represented in XML, but document types vary widely,

creating diverse challenges for high-volume publishing requirements. Documents can possess data that ranges from highly consistent and repetitive to extremely diverse content structures that defy accurate digital representation. For example, accident reports, product catalogs, employment applications, financial forms, and other types of documents are very amenable to automated XML conversion solutions. On the other hand, dissertations, marketing reports, résumés, news articles, and other documents feature abstract intellectual information with highly diverse components and inconsistent composition.

For highly structured content, identifying and tagging variables can easily be automated through forms, scripts, and other techniques, but mixed-structure data requires an XML authoring tool or a postauthoring conversion process. The assignment of tags to elements in a document is fundamentally a separate and distinct exercise from the authoring process. Authors may intuitively recognize elements – such as a “phone number,” “chapter heading,” “ingredient,” or “customer type,” but their identification as an element requires a start tag, an element type, and an end tag delimited by brackets (<phonenumber>). This process can be simplified and accelerated, but it can’t be eliminated.

Requiring content creators – knowledge workers such as technical writers, paralegals, insurance adjusters, law enforcement personnel, research professionals, and lab technicians – to perform manual tagging on original content is an unrealistic expectation in many environ-

ments. While considerable advancements have been made in XML authoring tools, they remain unappealing to many users who prefer standard word processing or desktop publishing software. Drop-down menus for tag selection, template support, advanced scripting, macros, and other modern enhancements to XML authoring systems still require manual tagging, an activity that is wholly separate from the document creation process.

In addition to the variability of document type, the thoroughness of document representation also may vary widely, depending on the application. It’s possible to categorize a document simply by title, date, subject, and author, or render a document with hundreds of variables. The wide variability in content types, content sources, and data applications requires the customization of virtually every high-volume XML publishing system in order to achieve specific business goals.

High-volume XML publishing also frequently involves legacy documents in paper or electronic form. It’s not uncommon for an XML publishing project to involve warehouses of paper in bankers’ boxes, thousands of pounds of microfilm and microfiche, thousands of tapes or disks in obsolete, proprietary formats, and/or literally terabytes of PDF files. In many industries the events that initiate an XML project – such as mergers/acquisitions, new document management procedures, government regulations, and/or new business initiatives – are also the events that involve the greatest volume of archival information requiring XML conversion.

Key Requirements for a Mass Conversion Platform

For the full benefits of XML to be realized for high-volume conversion of

AUTHOR BIO

Evan Huang is cofounder and chief technology officer of XMLCities, a developer of XML content creation, conversion, and publishing tools. A frequent contributor to various technical journals, Evan previously held positions at SRI and Adobe Systems and taught at Notre Dame and Northwestern Polytechnic. He holds a PhD in electrical engineering from Notre Dame.

JAVA™ in JUNE

Especially in New York

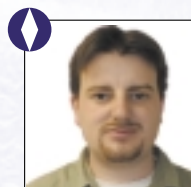
JDJEDGE

conference & expo

INTERNATIONAL JAVA DEVELOPER CONFERENCE & EXPO

A Sampling of Java-Focused Sessions

- Java 1.4: What's New?
- Building Truly Portable J2EE Applications
- J2ME: MIDlets for the Masses
- WebScripting Technologies: JSP/CFML/Velocity
- Where Is Swing in This New Web Services World?



Alan Williamson
Java Chair
Editor-in-Chief
Java Developer's Journal

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

OCTOBER 1-3
SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

Featuring...

- Unmatched Keynotes and Faculty
- The Largest Independent Java, Web Services, and XML Expos
- An Unparalleled Opportunity to Network with over 5,000 i-Technology Professionals

Who Should Attend...

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



JACOB K. JAVITS CONVENTION CENTER, NEW YORK, NY



FOCUS ON JAVA

Java, now mainstream, is the dominant back-end technology upon which next-generation technologies are evolving. Hear from the leading minds in Java how this mainstream technology offers robust solutions to i-technology professionals and senior IT/IS strategy decision makers.

The Java Fast Paths on June 24, a Java-focused CEO Keynote Panel, and comprehensive conference sessions will focus you on Java every information-packed day!

Register Online Today
for Lowest Conference Rates
Early Sell-Out Guaranteed!
www.sys-con.com

FOR MORE INFORMATION

SYS-CON EVENTS, INC. Michael Pesick
135 Chestnut Ridge Rd. Michael@sys-con.com
Montvale, NJ 07645 201 802-3057

Owned & Produced by:

SYS-CON
MEDIA

SYS-CON
EVENTS

Sponsored by:

IONA | END 2 ANYWHERE™ | TogetherSoft | bea | ADOS

SilverStream | MERANT | Actional | PolarLake

Media Sponsors:

Charles F. Goldfarb's XML TIMES.com | Federal Computer Week | WebServicesMail | j2ee

WRXK | SDTimes | BASIS | Webservices.org | Java Skyline

WebServices JOURNAL | JAVA DEVELOPERS JOURNAL | XML JOURNAL | wireless BUSINESS TECHNOLOGY

WebLogic DEVELOPER'S JOURNAL | ColdFusion DEVELOPERS JOURNAL | WebSphere DEVELOPERS JOURNAL | CF Advisor

Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. SYS-CON Publications, Inc., is independent of Sun Microsystems, Inc. All brand and product names used on these pages are trade names, service marks, or trademarks of their respective companies.

unstructured content, a production platform will increasingly use intelligent automated solutions (autotagging) to achieve productivity and quality control objectives. In some applications these automated tagging solutions can coexist with scripts and templates, but the ultimate solution should be driven by content goals. Content authors will resist using XML tools or a template in many environments; consequently, content is often generated outside an organization where conformance to standard authoring procedures is impossible.

Another requirement for high-volume XML publishing is that the DTD/Schema structure should be determined by the user or application, not by the XML editing tool, production platform, or skills of the operators. In addition to high cost and long cycle time, outsourcing XML conversion also frequently compels the use of standardized or simplified DTDs for productivity reasons, rather than the richly structured DTDs demanded by the application. Sacrificing powerful DTDs to reach cost or productivity goals is a short-term strategy that may negatively impact the overriding knowledge-management goals of the organization.

XML publishing platforms also require an integrated system that addresses all steps in the conversion process, from input through tagging, proofing, validation, and quality control. To accomplish this in an efficient manner, high-volume XML publishing requires batch processing and workflow management solutions that optimize productivity.

Automated Markup

New technologies now emerging replace manually intensive markup processes with automated tagging to enable efficient in-house conversion projects as well as support outsourcing solutions. Autotagging technology will assign data into appropriate DTD tags based on comprehensive rules or identified strings set up by content administrators. The amount of markup that can be accomplished with these new technologies will depend on the type of document converted and the resolution of the DTD structure assigned. With many types of content, over 95% of the markup can be accomplished with autotagging, enabling mass conversion projects to be accomplished efficiently without outsourcing to a service bureau.

These new automated markup tools allow content administrators to develop comprehensive rules that define, identify, and assign element tags based on user-specified DTD/Schemas. These rules can be extremely sophisticated

patterns set up with strings based on key words, phrases, document location, data type (numeric, alphanumeric), or other identifiable pattern or identifier.

Representative sample documents that are fully marked up are used to identify the patterns, signifiers, and rules that indicate element tags. Drop-down menus list the possible rule components, such as key words, digits, spacing, and formats that may be grouped to form a rule.

Automated markup can also be extremely effective for converting simple DTD/Schemas into complex DTD/Schemas and in replacing costly, cumbersome, scripted approaches to XML conversions. For highly variable content with a thorough DTD, 60–70% of the markup can be accomplished – enough to make an enormous impact on the cycle time and cost of XML publishing.

Workflow Design and Production Control

High-volume XML publishing requires workflow management solutions to achieve appropriate productivity and production control goals. For enterprise-scale processing of XML data, content administrators want to assign and distribute separate production tasks to various operators to maximize output, balance workloads, and ensure the highest quality. In addition, the workflow design must provide effective management oversight, enabling appropriate monitoring, notification, approvals, and audit trails. In most applications the production system will also require policies and procedures to ensure appropriate information security and workstation statistics for production control. To accomplish these objectives, workflow management tools need to be highly integrated into the mass conversion process from authoring and OCR scanning, through a multistep markup process and on to proofing and validation.

To achieve this high level of integration between workflow management and production tasks, XML publishing platforms require a project dispatching system from initial input (electronic document or scan), autotagging, manual tagging, proofing, validation, and posting. The system should manage this process by automatically dispatching work-in-process files to workstations based on a predetermined design established by administrators. The workflow dispatching system is integrated into the scanning, autotagging, manual tagging, and proofing/validation tools throughout, and work-in-process status can be determined at any time by authorized content managers.

Legacy document conversion often must be accommodated in mass conversion operations. Capturing accurate data from scanned documents is conducted as a preprocess module prior to XML conversion and may require varying degrees of proofing/quality control. Conversion of legacy digital file formats such as MS Office Suite files, PageMaker files, RTF files, or PDF files into XML requires less proofing/quality control process prior to XML markup and also can be seamlessly integrated into the mass conversion process.

Workflow design becomes an important factor in optimizing XML conversion. The number of workstations for any particular production process will be determined by the specific nature of the project, such as expected accuracy of the OCR process or quantity of document types (i.e., scanned or electronic documents). Workflow design using automated markup processes will be significantly different from those dependent on manual markups. In manual markup processes workflow design is typically based on DTD complexity where specialized tagging tasks are distributed to operators specifically trained in a subset of the content. Automated markup processes using autotagging are designed to eliminate these specialized tagging processes, enabling highly sophisticated markups to be accomplished through only two stages: autotagging and manual tagging for exception handling. In this way autotagging allows the workflow design to be simplified and fixed across a variety of conversion projects; the number of workstations will be dictated primarily by volume.

The combination of autotagging solutions with effective manual markups for exception handling and quality control allows content administrators to specify the depth and detail of the DTD/Schema based on the application without a loss in productivity or cycle time. To allow for breaking up the tagging process for complex documents in a production setup, advanced DTD editors provide multiple “views” of a richly structured DTD tree, further simplifying and accelerating the conversion of specialized documents.

Designing a High-Volume Conversion Process

The real-world variability in content types, content sources, volume, and in-house resources requires customized process solutions to realize efficient high-volume conversion. Only through a rigorous analysis phase and thorough process design can an XML conversion system be optimized for any particular environment. The use of consultants without

XML-NextG of Enterprise Deployment™

XMLEDGE
conference & expo

INTERNATIONAL XML CONFERENCE & EXPO



Norbert Mikula
XML Chair
Board of Directors, OASIS
Industry Editor
Web Services Journal

A Sampling of XML-Focused Sessions

- XML Web Services: Standards Update
- Using XML for Rapid Application Development and Deployment with Web Services
- Unlocking the Promise of XML: From Hype to How-To
- The Use of XML Technologies to Enhance Security
- Content Management with XML

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

OCTOBER 1-3
SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

Featuring...

- Unmatched Keynotes and Faculty
- The Largest Independent XML, Java, and Web Services Expos
- An Unparalleled Opportunity to Network with over 5,000 i-Technology Professionals

Who Should Attend...

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



JACOB K. JAVITS CONVENTION CENTER, NEW YORK, NY



FOCUS ON XML

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

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

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

Register Online Today
for Lowest Conference Rates
Early Sell-Out Guaranteed!
www.sys-con.com

FOR MORE INFORMATION

SYS-CON EVENTS, INC. Richard Anderson
135 Chestnut Ridge Rd. Richard@sys-con.com
Montvale, NJ 07645 201 802-3056

Owned & Produced by:

SYS-CON
MEDIA

SYS-CON
EVENTS

Sponsored by:



Media Sponsors:



commercial ties to specific vendors is often extremely useful in fully evaluating options and understanding the critical links between technology tools and human-factor organizational issues.

The process for mass conversion typically begins with a planning phase that addresses variables such as source material evaluation, target evaluation (number of targets and DTDs), volume estimates, and time frame. These issues will define the project scope, budget, and quality levels. The design phase will determine process flows, organizational requirements, infrastructure needs, workload balancing tools, and other implementation requirements.

Because the accuracy of OCR engines, organizational issues, and XML conversion quality must be thoroughly validated before full production, a proof of concept is generally a part of any comprehensive

implementation. Output quality of every document source must be scrutinized in detail to eliminate errors and anomalies. In addition, productivity goals, cost estimates, and other objectives require validation.

Even with a rigorous testing phase, ongoing monitoring of quality will be required due to the variability of labor-intensive proofing and tagging steps. Conversion schedules, material trafficking, exception reporting, and delivery mechanisms need to be optimized after system rollout to achieve full benefits and efficiencies.

As XML publishing proliferates in corporations, governments, educational institutions, service bureaus, and other organizations, seamlessly integrating the markup process into content generation and authoring will be a primary objective for tool and platform developers. For cycle time and information

security reasons, organizations will increasingly look toward cost-effective methods to accomplish the markup process through in-house means.

Trends in both workflow management and automated markups anticipate an expanded role for XML content administrators. Rather than specific expertise in XML syntax and semantics, content administrators will require new skill sets in organizational leadership, process design, and process management. The widespread acceptance of XML – along with advances in tools and publishing platforms – will usher in a new, expanded role for the content administrator in today's dynamic organizations.

Pattern Recognition for Automated XML Tagging

The inherent complexities and inefficiencies of manual markup tagging for mixed-structure content is being addressed through tools that use rules driven by identified strings and patterns to automate the tagging process. These tools create unique pattern files that are associated with specific elements. Elements generally have multiple rules or patterns that can be used to identify accurate tags.

Rather than writing Perl scripts, drag-and-drop is the fastest and most accurate way to associate structure points and data to a pattern. Sample documents are fully marked up and used to identify the patterns, signifiers, and rules that indicate element tags. Common expressions that may indicate relevant text and structure associated with a rule can be quickly identified using color coding, multiple windows, and other user interface techniques. Drop-down menus list the possible rule components such as key words, digits, spacing, and formats that may be grouped to form a rule.

Testing, refining, and optimizing the rules over sample documents are integral parts of the process. Processing sample documents identifies additional rules, exceptions, anomalies, and other factors that can be accounted for in the full production run. Rule libraries can be developed to allow content administrators to quickly build autotagging processes for common elements over a wider variety of document types. In this way the automated-tagging process can be optimized over time, eliminating a vast majority of the manual markup requirements for XML publishing.

Practical Examples

Using the conventions for pattern components and character mnemonics in Tables 1 and 2, a sample pattern file can be created. Table 1 summarizes the

ITEM	DESCRIPTION
<rules.list>	The Rules List holds the entire Pattern file. It contains one or more pattern items.
<pattern>	Each pattern performs a search according to the criteria in its blocks, and limited to the section it defines with <include> and <exclude>.
<block>	Block is used to break the <pattern> into pieces. Each piece may then be associated with an element.
<char-set>	Char-set is used to represent a set of characters (that are allowed at that point). It checks one character at a time. It is the equivalent of the Perl character class.
<list>	List is used to enumerate possible choices. For example, to find a month a <list> might include "January", "February", etc.

TABLE 1 Major pattern components

DROP-DOWN MENU ITEM	APPLIES TO	PERL EQUIVALENT	EXPLANATION
any.letter	<block> <char-set>	[a-zA-Z]	"a", "b", "A"...
any.lower	<block> <char-set>	[a-z]	"a", "b", "c"...
any.upper	<block> <char-set>	[A-Z]	"A", "B", "C"...
letter.or.digit	<block> <char-set>	[a-zA-Z0-9]	"a", "b", "1", "2"
any.digit	<block> <char-set>	[0-9]	"1", "2", "3"
whitespace	<block> <char-set>	\s	Space, tab, or new-line character
one.space	<block> <char-set>		" " – Space character
tab	<block> <char-set>	\t	Tab character
new.line	<block> <char-set>	\n	New-line character
any.but.new.line	<block> <char-set>	.	Any character except new-line

TABLE 2 Character mnemonics

major pattern components required for effective automated markup. The elements are used to initiate and execute a pattern search. Drop-down menus facilitate rule building. Table 2 includes some of the character mnemonics and their relationship to Perl scripts.


A rule for identifying an element begins with an examination of what patterns constitute the data. Refer to the date January 1, 2002, as seen below, for an example.

The process for creating a rule based on this character set requires defining the pattern for each component (month, day, comma, year) and associating the block to the DTD. Each pattern is created through drag-and-drop by simply highlighting the component with a mouse click. The completed pattern file, composed of multiple patterns, would then look like Listing 1.

The pattern in Listing 1 would produce the following in the XML output:

```
<document>
<date>
```

```
<month>January</month>
<day>1</day>
<year>2002</year>
</date>
</document>
```

Common rules, such as the one above, can be stored in rules libraries and reused for multiple document types. Many observers believe that only through such automated markup techniques can XML publishing be efficiently incorporated in high-volume content applications consisting of mixed-structure content. Further developments in the use of intelligent rule-based algorithms to automate the markup process will emerge for applications in the legal, financial, technical, medical, regulatory, and other fields involving mixed-structure content. These developments will involve both Boolean operations and other forms of artificial intelligence, such as neural networks, to produce XML markups. 

CTO @ XMLCITIES.COM

LISTING 1

```
<?xml version='1.0' encoding='ISO-8859-1'
standalone='yes' ?>
<rules.list>
<pattern>
<include element="all"/>
<block element=
"document/date/month/+DATA">
<list>
<li>January</li>
<li>December</li>
<!-- list all months here -->
</list>
</block>
<block element="document/date/day/+DATA">
<char-set max-count="2" min-count="1">
<any.digit/>
</char-set>
</block>
<block><string>,
</string>
</block>
<block element=
"document/date/year/+DATA">
<char-set max-count="4" min-count="2">
<any.digit/>
</char-set>
</block>
</pattern>
</rules.list>
```

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

January 1, 2002

A month

A space

1 or 2 digits

A comma and
a space

2 to 4 digits

EnginData Presents

2002 Developer Market Survey Reports



engindata.com

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

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

engindata ✓
RESEARCH

*The only real standard that's mandatory is TCP/IP —
that aside, may the best new standards win!*

Markup is Madness

I've been reading that the first Web page went online 10 years ago. I remember what it was like being involved with software development at that time. My initial feeling was that using markup in a browser was less than optimal because it leaves too much of the presentation details up to the browser, rather than allowing developers to specify absolute coordinates and other necessary attributes for various objects on the page.

After doing much in the way of Web development and design over the past 10 years, I still agree with myself. Markup sucks for anything other than attributing style changes and links in text copy. For this purpose it's arguably easier than keeping track of style runs and keeping them in sync with text as it's being edited.

Markup is messy. It's hard to work with — staring at it for more than a few minutes gives me a headache. Manually coming up with HTML is a tedious exercise in trying something, checking out how it looks in all of the various browsers you support, and continuing your iterations. Not all browsers speak exactly the same dialect or have the same idea of what you'd like to see. Using a good WYSIWYG HTML editor like Adobe GoLive (which produces very nice markup) that generates code compatible across browser versions you select is a good companion to hand-editing HTML. Some of my friends refuse to use an editor, but I can't fathom how long it would take to generate by hand some of the more layout and graphically intense sites expected today.

I used to work with some SGML proponents who were thrilled that they could generate HTML and PostScript versions of their source documents with ease. I wonder how much of their time was spent on figuring out which tags to type instead of improving the clarity and focus of their content. Honestly, it seems like PDF is a better standard for platform and resolution-independent document authoring; what you see on the screen is exactly what you'll see on the printer. Apparently Apple agrees, as PDF is one of the native rendering types in its new

Quartz graphics engine. Every application on Mac OS X, when printing, generates a PDF document that's rendered by the printer. Very nice. Regardless, PDF is even harder to write by hand than markup, and not a good choice for the kinds of interactive user experiences that people expect to have. Flash is pretty neat, but not the best choice either. Luckily, HTML, PDF and Flash can all be viewed simply as renderable media types; they can be embedded or framed by better technologies.

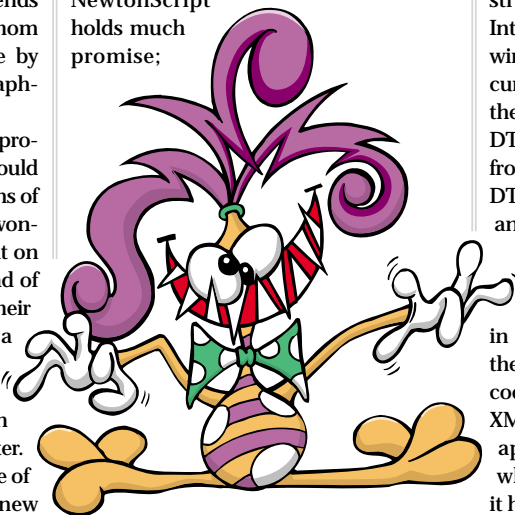
JavaScript or Apple's NewtonScript are neat for specifying online experiences. Note I don't say "Web pages." I believe that what people really want on the Internet is a vast, interconnected, explorable, media-rich user experience where all users can design and build objects and new experiences from within the browsing application itself. The dynamic nature of JavaScript and NewtonScript holds much promise;

both sit on a foundation of dynamic objects called *frames*. A frame is an unordered list of named values that can be strings, floating point numbers, arrays, raw data, or other frames. Apple used this methodology in very nice, new, and exciting ways in its ill-fated Newton series of PDAs in the early 1990s. There's no reason that user-interface development has to be done with strongly typed languages like C and C++ or even Java; most of the logic in building user-interface components and views works very nicely with dynamic interpreted languages with prototype-based inheritance. Scripting.

The markup folks want to extend their domination to the data space. They tell me that XML is the way to go. To introduce a new data type to the world, all you have to do is draw up a cryptic DTD that defines the rules for the data and then generate data files, all using markup tags. I see problems with this for many reasons. Markup text is larger and more verbose than it needs to be. Too much information is revealed about the data and its structure via the DTD, and as most Internet communication goes over the wire unencrypted, this seems pretty insecure. It's also simply too much of a pain in the butt, and way too static to define a DTD for arbitrary data you want to move from point A to point B (I realize that DTDs are optional). XML may be a good answer for configuration or settings files, or for enterprise data exchange in commerce applications, but I don't see developers sticking all their data in tree structures, generating XML from them, passing them to functions in their code that just go ahead and parse the XML back into a tree structure — all to appease the proponents of markup; while XML seems like a neat idea, so far it hasn't lived up to the hype.

AUTHOR BIO

Steve Klingsporn is a software designer and developer who specializes in desktop and portable user interface and server-side Web development.



Web Services Skills, Strategy, and Vision

web services **EDGE**
conference & expo

INTERNATIONAL WEB SERVICES CONFERENCE & EXPO



Sean Rhody
Conference Tech Chair
Web Services Track Chair
Editor-in-Chief
Web Services Journal

A Sampling of Web Services-Focused Sessions

- Starting Out in Web Services: Fundamentals of Web Services (WSDL, UDDI, SOAP)
- Exploring .NET myServices Initiative
- Standards Watch: Reviews and Discussions of the Interactions of All the Relevant Standards
- Guarding the Castle: Security and Web Services
- The Microsoft Way: .NET, Passport, and other MS Technologies for Web Services
- Laying Down the Rules: Web Services and Business Process Engines
- Practical Experiences with Web Services and J2EE
- The Odd Couple: Making .NET and J2EE Work Together

JUNE 24-27

**JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY**

OCTOBER 1-3

**SAN JOSE
CONVENTION CENTER
SAN JOSE, CA**

Featuring...

- Unmatched Keynotes and Faculty
- 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 /C Level Executives
- Analysts, Consultants

Sponsored by:



Media Sponsors:



**SYS-CON
MEDIA**

Owned & Produced by:

**SYS-CON
EVENTS**



JACOB K. JAVITS CONVENTION CENTER, NEW YORK, NY



FOCUS ON WEB SERVICES

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

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

Take home the tools and resources you need to keep the competitive edge. Web Services will come into clear focus at Web Services Edge 2002 East. Don't miss it!

Register Online Today
for Lowest Conference Rates
Early Sell-Out Guaranteed!
www.sys-con.com

FOR MORE INFORMATION

SYS-CON EVENTS, INC
135 Chestnut Ridge Rd.
Montvale, NJ 07645

WEST:

Richard Anderson
Richard@sys-con.com
201-802-3056

EAST:

Michael Pesick
Michael@sys-con.com
201-802-3057

This is one of the "property list" XML-based preference files for Mac OS X:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM
"file://localhost/System/Library/DTDs/PropertyList.dtd">
<plist version="0.9">
<dict>
  <key>AutoHideTime</key>
  <real>1.0000000000000000e+01</real>
  <key>DisplayBehind</key>
  <false/>
  <key>LineLimit</key>
  <real>1.0000000000000000e+03</real>
  <key>LogFiles</key>
  <array>
    <string>/var/tmp/console.log</string>
  </array>
</dict>
</plist>
```

Here's what the same data would look like in Newton frame format:

```
{
  autoHideTime: 1.0000000000000000e+01,
  displayBehind: false,
  lineLimit: 1.0000000000000000e+03,
  logFiles: [ "/var/tmp/console.log" ]
}
```

Obviously, there's little win in terms

of readability, ease of authoring, or space considerations in the XML approach. Apple essentially developed a "PropertyList" DTD/XML format for what it pioneered with Newton frames (JavaScript is derived from NewtonScript) in the early 1990s. The result is illegible, but standard! The frame example doesn't require typed data; the XML example has tags that define the types. If you were building a system, which would you choose? Would you bet your company on it? What are the drawbacks? How much time will you spend worrying about the underlying technology itself instead of the problem at hand?

There's much talk about where the Web will be in 10 years. I hope that, by then, the Web is mostly markup-free. While I acknowledged above that marking up style information and links is a perfectly valid use of markup, the Web and the bloated-and-buggy state of today's browsers are glaring proof that using markup is a tedious and resource-intensive solution to a much simpler, and already solved, problem. And don't even get me started about how inefficient it is to open and tear down a socket connection for every resource on the page! Some form of package format for a page and all of

its associated resources would do much in the way of sparing network resources, speeding up the time it takes to download a page, and providing a nicer and more appealing user experience.

I encourage you to take a look at alternate technologies like Newton and JavaScript and work on fun experiments of next-generation browsers and networked user experiences as I've been doing. The Web came out of nowhere as the fruit of a handful of individuals' hard work and perseverance; I imagine that the next-generation Web technologies will spring from the same type of entrepreneurial origins. Luckily for all of us, there's plenty of room for experimentation and innovation, and all of these solutions can live parallel to each other. We have whatever tools we like or can create at our disposal, and the only real standard that's mandatory is TCP/IP. May the best new standards win! ☺

STEVE @ SEAPOD.ORG

Happy 4th Birthday, XML!



As the publisher of *XML-Journal*, we had to ask: What would be the best birthday present that anyone could give to XML?

From PAUL PRESCOD, an independent implementer of markup-based systems, a vocal advocate and critic of various technologies, coauthor of the *XML Handbook*, and an invited expert in the original XML standardization process:

I'd wish for XML to come back to the forefront. Rather than having huge wars about its associated technologies, it would be good if people would define XML vocabularies in their favorite schema language and see what works and what doesn't. From our experience in the SGML world we know that this is a challenging process that will take years to shake itself out. SOAP, WSDL, RDF, topic maps, etc., don't make that process go away.

And while they can be helpful, the current focus on them distracts the people who should focus on *their* XML vocabularies and not on how they relate to the clashes of the titans.

I would also wish for clarity in the schema world. Schemas matter. Having several schema languages makes communication among XML developers difficult. This leads people back to concentrating on the ancillary specifications rather than on the one thing that matters: the right configuration of XML elements and attributes that will allow organizations and individuals to interchange information that matters to them.

—Paul Prescod

From SIMON ST. LAURENT, an associate editor at publisher O'Reilly & Associates, focusing on XML, and the author of *XML: A Primer*, *XML Elements of Style*, *Programming Web Services with XML-RPC*, and *Cookies*:

The best birthday present might be to remember what the XML acronym stands for: Extensible Markup Language. *Extensibility* means enormous capability and diversity. It might be nice to start building software that takes advantage of that potential, rather than trying to squash everything into fixed vocabularies.

Markup means we're dealing with marked-up text here, not necessarily serialized objects or relational tables. There's an enormous amount of potential in that textual nature that's been largely forgotten in the rush to apply XML to systems integration.

And *Language* suggests that maybe XML is a coherent whole – that whatever vocabulary and structures you happen to use in your XML documents, you're building on XML. Thinking of XML as a whole rather than as a toolkit for building fragmented languages seems like it might open a lot of opportunities that are currently ignored.

Failing that, some peace and quiet would be nice....

—Simon St. Laurent



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



WebSphereDevelopersJournal.com

WebSphere
DEVELOPER'S JOURNAL



Introductory Charter Subscription

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

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Do You Have Access to the Internet?

The World's
Leading
Independent
WebSphere
Developer
Resource

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

Once you're in it...



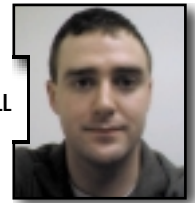
...reprint it!

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

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



RePrints



Yes, size matters

XMLWithoutWires

Part 1 of 2

Wireless transmission is becoming more and more common for many document types, and XML is no exception. But XML itself presents a number of challenges to the wireless medium. This article, the first in a two-part series, describes these challenges, and describes techniques that can be used to tailor an XML-aware system to the wireless world.

Wireless bandwidth continues to be at a premium, despite the long-standing promises of 2.5G and 3G networks. XML isn't binary, and while this means that knotty problems like little-endian/big-endian confusion are avoided, it also means that raw XML data isn't ideal for wireless transmission. In fact, design goal No. 10 for XML states that "terse-ness in XML markup is of minimal importance." This doesn't rule out XML for wireless transmission, but it does mean that compression is a must.

WBXML – Binary XML

The Wireless Application Protocol is a specification for developing applications that operate over wireless communication networks. The development of WAP is overseen by the WAP Forum, an industry consortium that includes Ericsson, IBM, Motorola, and Openwave. The wireless market is growing very quickly and is expanding to reach new customers with new services. To enable network operators and handset manufacturers to meet the challenge of deploying advanced services, WAP defines a set of protocols in transport, session, and application layers.

One component of the WAP specification that has been submitted to the World Wide Web Consortium is a description of Wireless Binary XML (WBXML for short). WBXML defines a compact binary representation of XML. It reduces the transmission size of XML documents, allowing more effective use of XML data on narrowband communication channels. WBXML achieves this by tokenizing XML files to reduce (1) the number of bytes needed to send data

over the wireless channel as well as (2) the need for a complex parser by constraining the file format so it can be rapidly parsed in binary form.

To send a WBXML file most efficiently, it's important to provide an initial token mapping for the channel. Some token mappings are specified in the WAP specification – WML, for example. Other XML formats may be used with WBXML without a mapping, but the transmission is more verbose, since the tag names need to be listed in a string list at the start of the transmission.

ASN.1

If all this talk of tokenizing files for binary transmission sounds familiar, it's because it reflects an older standard, Abstract Syntax Notation 1, published jointly by the International Standards Organization, the International Electrotechnical Commission, and the International Telecommunications Union. It hasn't gone away.

Like WBXML, ASN.1 is a collection of techniques and notations for specifying data structures in a machine-independent manner. Token sets defined using ASN.1 can be used in combination with standardized encoding rules such as the Basic Encoding Rules (BER) and the Packed Encoding Rules (PER). ASN.1 is supported by mature tools for encoding and decoding binary information into and out of BER and PER formats, as well as to other proprietary encoding formats.

ASN.1 is widely deployed in the wireless and fixed-line communications industry as an enabling technology for cell phone messages, radio paging, free

and premium phone services, and ISDN telephony. In addition, it is used for streaming audio and video over the Internet, and is fundamental to ISO security and messaging standards, including X.509 digital certificates and X.400 e-mail.

If a widely used protocol like ASN.1 already exists and has widespread adoption, then why generate a new protocol like WBXML especially for wireless adoption? This question has been posed in the past for many wireless initiatives, including WTLS (versus SSL), WScript (versus JavaScript), and, of course, WML (versus HTML). One answer in this case is that WBXML is specifically linked to XML and preserves the treelike nature of an XML document, allowing a recipient system to deal selectively with XML tags. ASN.1 and XML do not have congruent data models, and so the translation between them is not a straightforward exercise.

Another possible drawback is the lack of freely available tools for ASN.1. By contrast, the XML world contains many free tools, such as xalan for XSL stylesheets and xerces for Document Object Model processing. However, if ASN.1 were to be recommended for wireless XML processing, free tools would likely come onto the scene quickly.

This incongruity of XML and ASN.1 is being addressed by two ITU/ISO initiatives that map XML to ASN.1 and vice versa. The first, AML (ASN.1 Markup Language), represents ASN.1 data types as XML, and is to be the ITU-T Recommendation X.693 and ISO/IEC 8825-4. The second maps XML Schemas to ASN.1 in such a way as to preserve all the information contained in the XML Schema definition. This will allow data defined as XML to take advantage of ASN.1 tools such as DER (to encrypt and digitally sign data, for example) and PER

AUTHOR BIO

Mark O'Neill, CTO of Vordel, oversees the development of Vordel's technical strategy and product development in the areas of XML and public key cryptography. Mark designed and implemented Internet EDI solutions for Ireland's largest EDI value-added network, Eirtrade Services Ltd. He holds a double-honors degree in mathematics and psychology from Trinity College Dublin and studied neural network modeling at Oxford.

JavaOne

<http://java.sun.com/javaone>

(to send data over radio, for example). This initiative is to be the ITU-T Recommendation X.694 and ISO/IEC 8825-5. Information about these two initiatives is available at <http://asn1.elibel.tm.fr/xml>.

The focus on XML Schemas rather than DTDs for these two ITU/ISO initiatives reflects the fact that Schemas are closer to ASN.1 than DTDs are. However, tools are available to map between the two; one example is the ASN.1/XML Translator, which is part of IBM's XML Security Suite (see www.trl.ibm.com/projects/xml/xss4j).

Encoding Techniques

Focusing on document size, let's compare the encoding formats we've discussed so far: ASN.1, WBXML, and plain XML.

First, let's use the DTD that follows to define a simple purchase order document.

```
<!ELEMENT PurchaseOrder (LineItem+)>
<!ELEMENT LineItem (Description, Quantity)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ATTLIST LineItem
  Department ( Food | Hardware | Pharmacy)
  #REQUIRED>
```

The next code snippet is an example of an XML document that conforms to this DTD.

```
<?xml version="1.0" ?>
<!DOCTYPE PurchaseOrder SYSTEM "PO.dtd">
<PurchaseOrder>
  <LineItem Department="Food">
    <Description>Cookies</Description>
    <Quantity>5</Quantity>
  </LineItem>
  <LineItem Department="Hardware">
```

```
<Description>Hammers</Description>
<Quantity>10</Quantity>
</LineItem>
</PurchaseOrder>
```

Now let's look at an ASN.1 specification for the same simple purchase order document. In the following code you can see that we're using the SEQUENCE, INTEGER, and ENUMERATED ASN.1 data types. Information about the full range of ASN.1 data types is available at <http://asn1.elibel.tm.fr/xml>.

```
PurchaseOrder ::= SEQUENCE OF LineItems
LineItem ::= SEQUENCE {
  Description VisibleString,
  Quantity INTEGER,
  department Department }
Department ::= ENUMERATED {
  Food(0), Hardware(1), Pharmacy(2) }
```

Using BER encoding, we see that the document is considerably smaller than the plain XML document in the second code snippet. Note that PER encoding is optimized for radio transmission and compresses the document still further:

```
10 18
1A 07 'C' 'o' 'o' 'k' 'i' 'e' 's'
02 02 05
0A 00 00
0A 01 01
1A 07 'H' 'a' 'm' 'm' 'e' 'r' 's'
02 02 0A
0A 01 01
0A 01 01
```

Now we come to the WBXML specification for the same document. WBXML caters specifically to XML, so we define tokens to represent XML elements and XML attributes, as follows:

Tag code space (Tag Name : Token) :

```
PurchaseOrder : 5
LineItem : 6
Description : 7
Quantity : 8
```

Attribute start tokens:

```
(Attribute: Attribute Value Prefix : Token)
Department : Food : 5
```

The following shows the purchase order data represented in WBXML encoding, using the tokens we defined above. WBXML uses string tables to list strings used in the XML document, and following this string table we see that all the data takes the form of tokens.

```
01 01 03
10 'C' 'o' 'o' 'k' 'i' 'e' 's' 00 'H' 'a' 'm' 'm' 'e'
'r' 's' 00
45 C5 07 83 00 05 05 01
45 C5 07 83 09 06 0A 01
01
```

So Many Wireless Devices, So Little Time

The limited display capabilities of mobile handsets has led to the need to optimize data views/formats for display on these devices. While these new developments have benefited the mobile user experience and have increased uptake of mobile applications, they have added greater complexity to the role of content providers. With the increasing numbers of different devices available for retrieval and display of information over the Internet comes the added headache of customizing the display of the data on each device. Content providers who don't offer easy

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

web services **EDGE**
world tour 2002

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

access to their information across multiple devices risk losing customers. What's suitable for one device may be unacceptable for another. However, the problem is that re-creating and customizing data outputs for different media and on different devices is very resource-intensive and may eventually become unmanageable.

Sun Translets: Two Steps Instead of Three

XSLT is a standard that addresses the problem of transformations between the many proprietary and industry-vertical XML vocabularies. It can also be used to format documents for output to different display devices (phone handsets, PDAs) when using XML as a base data storage format.

XSL Transformations are executed using an XSLT engine, which typically tends to be a large and complex program. It reads in an XML file, reads in an XSL stylesheet, loads the XSLT logic into memory, and then applies the XSLT logic to the XML document. This three-step process is often a bottleneck for wireless deployment to multiple devices, losing valuable time for content delivery.

Sun's XSLT compiler simplifies the transformation process. First, a Java class known as a *translet* is produced from an XSLT stylesheet. This translet is then applied to the XML files that use the original XSL stylesheet. This technique has important advantages of speed and size over the traditional three-step method of applying stylesheets. The speed advantage is that step 1 – compiling the stylesheet into a Java class – only has to be performed once. The compiled translet can then

be applied to multiple XML documents.

The size advantage is that only the XSL logic used by the stylesheet is compiled into the translet. The rest of the XSL language logic isn't needed. For most stylesheets this reduces the size of the application logic considerably. XSLT engines weigh in at 700 to 800KB – translets may be only 10 to 50KB. Sun has donated their translet engine to the open-source community, with the result that the xalan XSLT engine now incorporates translet logic. The speed gains from using translets make XSL more palatable as a technology choice to address the problem of content deployment to multiple wireless devices.

The smaller size of translets, compared to using a full XSLT engine, means that this is suitable for wireless devices that incorporate J2ME (Java 2 Micro Edition) implementations. Sun's J2ME specification defines a Connected Limited Device Configuration (CLDC) that can be implemented by a handheld wireless device. A CLDC device may well have a processor of limited prowess (in comparison to a desktop system) and a memory footprint of between 128 and 512KB. This means that running a full XSLT engine is out of the question. However, a translet is much smaller and can run in the tight memory confines of a CLDC device.

The Downside of XSL

Notwithstanding the efficiency gained when using translets, XSL can be an unappealing development tool because of the complexity inherent in creating and maintaining XSL stylesheets. XSL is a templatic (template-based) language

that can confound software developers who are used to the logic in their programs being executed in a top-to-bottom sequence. The fact that the templates in an XSL stylesheet can be jumbled up into a different order, yet the stylesheet still produces the same output, is counterintuitive.

The complexity of the XSL language can result in solutions that are difficult to maintain. Adding to this problem is that although XSL doesn't mix presentation with data, it does mix presentation with logic. As mentioned above, logic expressed in XSL can be complex. XSL skills are required by a Web designer who wishes to change the graphical output of a transformation. Technologies such as JSP (JavaServer Pages) or ASP (Active Server Pages) are more similar to traditional programming languages and as such may be more suitable for ongoing maintenance and code reuse.

Summary

In this article I've discussed the compression techniques for XML that are essential for the wireless transmission of XML documents. I also covered the use of XSL, with and without translet technology, to tailor XML base data for multiple wireless devices. In Part 2 I'll discuss other technologies for wireless XML, such as VoiceXML, WAP and WML, and HDML, as well as wireless techniques for signing and encrypting XML documents. ☛

MARK ONEILL @ VORDEL.COM

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

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

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



PRESENTERS...

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



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

EXCLUSIVELY SPONSORED BY



BOSTON, MA (Boston Marriott Newton) **JANUARY 29**
WASHINGTON, DC (Tysons Corner Marriott) **FEBRUARY 26**
NEW YORK, NY (Doubletree Guest Suites) **MARCH 19**
SAN FRANCISCO, CA (Marriott San Francisco) **APRIL 22**

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

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





AN INTRODUCTION BY EXAMPLE

A build tool based on XML, using Java classes

A n n B l a c k & Y u T a n g

Ant is a build tool that provides a platform-independent alternative to shell script-based build tools like make. Developed using Java classes and based on XML, Ant preserves the “write once, run anywhere” Java motto. With Ant’s plethora of built-in tasks and easy-to-follow documentation, it doesn’t take an Ant guru to begin building with it.



This article introduces the basic concepts of Ant and examines its built-in capabilities that can be used when building and testing J2EE applications in WebSphere. We also demonstrate how to extend Ant's function to interact with IBM WebSphere Application Server.

Ant and XML

The build process for an Ant user's project is defined through an XML buildfile. Since the buildfile is constructed with a common standard, it isn't necessary to learn – and adhere to – a new build tool syntax. XML also makes the buildfile very readable and self-explanatory. This helps with the learning curve when you're getting up to speed with the project's build process.

Ant can be bundled easily with a software installation since it's supported on all Java-supported platforms and has a simple installation process. For example, code samples and tutorials that accompany application server software can also have Ant buildfiles to simplify their build-and-deploy process.

Ant is task-driven, with each task implemented by a Java class. Since it's based on Java, it's no longer necessary to maintain a buildfile per platform. Ant's configuration file is constructed once and run on any platform.

Ant provides a set of predefined tasks that make it quick and easy to get started. However, Ant isn't limited to these tasks: it's extensible through custom Java classes that implement a common interface. Not only are the tasks extensible, but Ant provides interfaces for creating custom build listeners.

A JAXP XML parser, along with a JDK (Java Developer's Kit), must be installed and configured on the machine that will be running Ant. If WebSphere Application Server is installed, then both the XML parser and the JDK are already configured. The Ant and the JAXP XML parser installables are downloadable at <http://jakarta.apache.org/ant/index.html>. Once downloaded and unpacked, only a few system variables remain to be set up before you can run Ant.

Running with Ant

When installation is complete, Ant is invoked by typing “ant” at a command prompt. Once invoked, Ant will locate a configuration file, `build.xml`, in the same directory from which it was invoked. By specifying the `-find` command-line parameter, Ant will attempt to locate the `build.xml` file from the current directory and, if not found, from each parent directory until the root directory has been searched. The configuration file isn't constrained to the name `build.xml`. Ant can be pointed to any configuration file using the `-buildfile <file_name>` command-line parameter.

The `build.xml` file is constructed with four main components (the relationship between them is displayed in Figure 1):

1. **Project:** Each buildfile has one *project*. The project is a unit of work that defines *target(s)*, which consist of *task(s)*. The project has three attributes: *name*, *default*, and *basedir*. Only the default attribute is required. The name attribute is a descriptive identifier for the project. The default attribute defines the name of the target that will be run by default when none is specified on the command line. The *basedir* attribute defines a directory from which all path calculations are made. If no *basedir* is specified, it defaults to the directory containing the buildfile. The *basedir* attribute can be overridden by a property with the name *basedir*.
2. **Target:** A target is a task or group of tasks that, when executed, do a particular job. Targets can be dependent on other targets, specified through the “depends” target attribute. This means that before the target is executed, Ant will execute each dependent target first, and, recursively, each of the dependent target's dependencies. After the dependencies have been executed, the tasks specified in the target will be invoked in the order in which they were defined.

3. **Task:** Each task corresponds to a Java class. A task can be a predefined Ant task, an optional Ant task, or a custom task. A task can have one or more attributes defined. These attributes can reference property values, which are resolved at runtime before the task is executed. All tasks have a name attribute that's used when Ant writes messages to a log.

4. **Property (optional):** A project can be associated with zero or more properties, which are name/value pairs or sets of properties from a file or resource that can be referenced in targets and tasks. The properties are defined via the property tag in the buildfile, or they can be defined through the command-line invocation of the buildfile. The value for the property provided at the command line will take precedence over the value specified in the buildfile. Additionally, Ant provides access to all of the system properties without having to define them as properties in the buildfile or on the command line.

Build.xml Example

Listing 1 is an example `build.xml` file. WebSphere Application Server AE and AEs v4.0 install with tutorials for learning the various aspects of application development on WebSphere (www.ibm.com/software/webservers/appserv/doc/v40/ae/index.html). In the advanced tutorial a CMP entity bean is assembled and deployed onto the WebSphere application server. For this example pretend the bean was altered and needs to be redeployed onto the server.

The project, named `CMP11`, defines the default target as `do_all`. When Ant is invoked without a specified target name, the `do_all` target is executed. This target doesn't define any tasks, but it does list dependencies. Before any of the tasks defined in the target are invoked, these dependencies will be invoked, and in the order in which they're listed: `cleanUp`, `init`, `preDeploy_ejb`, `ejbDeploy`, and, finally, `hotDeploy`.

Referenced throughout the project's targets and tasks are several properties that have been defined for this build. The properties define directory paths and names once, rather than in multiple locations throughout the project's targets and tasks. This minimizes the number of changes required when the source files are moved or when names are altered.

Notice that the `hdir` property makes a reference to the “`WAS_HOME`” property. While the property isn't defined in the buildfile, it is passed in on the command line via the `-D<property_name>` parameter so a system environment variable can be used to set it (see Figure 2). The last property defined uses the environment attribute of the property tag. This property is assigned the name “system” and allows all of the system environment variables to be accessed by system. `<environment_var_name>` in the targets and tasks.

The `cleanUp` target uses Ant predefined tasks to remove any directories and files that may be left over from previous runs of the build. The `init` target sets up the file system for the remainder of the build. It also uses several of Ant's predefined file system tasks such as `copy` and `mkdir` to manipulate the directories and files to initialize the project.

The `preDeploy_ejb` target consists of two tasks: compiling the source files and jarring the resulting class files with their deployment descriptor. It is invoked by Ant after the `init` target has completed. The first task specified in this target compiles the source files into a build directory that was created by the `init` target. The second task jars these compiled files into a JAR directory created by the `init` target. Since the `preDeploy_ejb` target relies on both of these directories being created by `init`, it has defined the `init` target as a dependency. Ant, however, is sophisticated enough to know that the `init` target was already executed as a dependency to the `do_all` target and therefore the `init` target is not invoked twice.

After the predeployment is finished, the `ejbDeploy` target is invoked. This target uses the `exec` task to invoke the `ejbDeploy` tool installed with WebSphere Application Server. This tool generates the deployment code for the specified `.jar` file – in this case the `.jar` file created during the pre-deployment. Since it generates deployed code from the `.jar` file created



in the `preDeploy_ejb` target, the `ejbDeploy` target has `preDeploy_ejb` as a dependency.

The final target to be invoked is the `hotDeploy` target, which specifies two custom Ant tasks and a predefined copy Ant task. The two custom tasks involve stopping and starting an enterprise module that's specified as a project property at the beginning of the buildfile. After stopping the module, the new deployed .jar file is copied into the specified installed application folder and the module is then restarted. The `hotDeploy` function of the WebSphere Application Server reloads the EJBs specified in the new deployed .jar file.

Built-in Ant Tasks and Structures

Ant provides a set of powerful, built-in tasks ready for use. We've seen some of them in the sample build.xml file. Let's take a closer look.

- **javac:** Searches the source directory recursively for Java source files and compiles them into Java class files in the target directory. The `javac` task compiles a source file only when there is no corresponding class file or the source file is newer than the class file.
- **exec:** Provides a way to call operating system commands within Ant. Users can specify which command to execute based on OS type. In the sample build.xml file, `ejbdeploy.sh` is invoked on non-Windows platforms while `ejbdeploy.bat` is invoked on Windows platforms.
- **fileset:** Not a task, rather an element that specifies a set of files. A `fileset` element can be nested in other tasks that operate on a selected set of files. A `fileset` element uses "includes" and "excludes" attributes to select files that (1) match a pattern, (2) don't match a pattern, or (3) match one pattern and not another. The patterns supported by Ant include '*', '?', and '***'. The character '*' is used to match any number of characters in the filename; '?', to match any one character in the filename; '***', to match any levels of directory in the complete path of the files. The following example `fileset` element specifies all XML files under the source directory with names that don't begin with "test":

```
<fileset name="non_test_xml" dir="${src}" id="xmls"
includes="**/*.xml"
excludes="**/test*.xml" />
```

- **copy:** Copies a file to another file or a set of files to a directory. When copying a set of files, the set is specified by using the nested `fileset` element(s). A file is copied only when the source file is newer than its target or the target doesn't exist. The following demonstrates the copy task:

```
<copy todir="${build}/META-INF">
<fileset dir="${source}/META-INF"/>
</copy>
```

- **jar:** Creates a .jar file. A user can include and/or exclude files from the .jar file via attributes of the `jar` task or nested `fileset` elements.

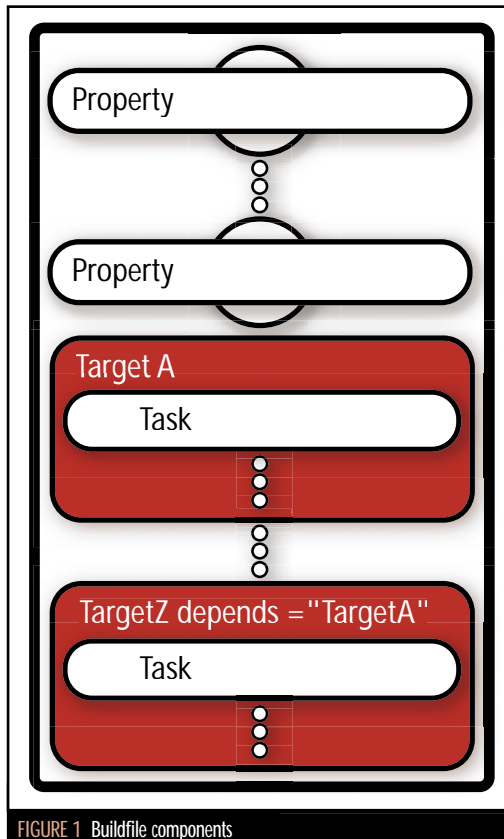


FIGURE 1 Buildfile components

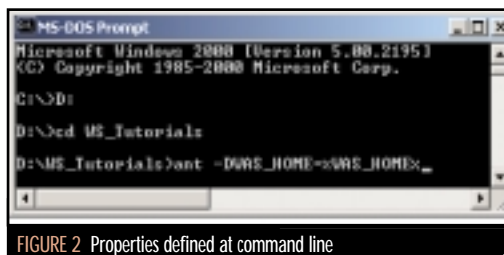


FIGURE 2 Properties defined at command line

The following example of the `jar` task creates a .jar file that includes all files specified in the nested `fileset`:

```
<jar jarfile="${dist}/${app_name}.jar">
<fileset dir="${build}/${app_name}"
includes="**/*.class" />
</jar>
```

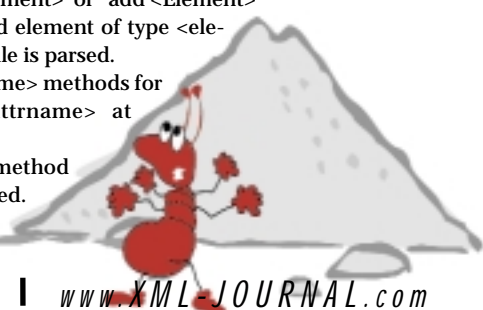
Extending Ant

Ant can be extended beyond built-in tasks with custom tasks and custom build listeners. Both are developed by extending the Ant framework. Let's investigate this in more detail.

Writing Custom Ant Tasks

Every task in Ant, including built-in tasks, is implemented as a Java class. Users can create a custom Ant task by creating a Java class that extends the class `org.apache.tools.ant.Task`. In addition, the following methods are usually required for simple tasks:

- **public void execute():** Throws `org.apache.tools.ant.BuildException`. The `execute` method needs to be overridden to perform the task.
 - **public void set<Attrname>(<attrType> attr):** If the custom task has attributes, a setter method is required for each attribute of the task. The setter method should be named 'set' <Attrname> where <Attrname> is the capitalized name of the attribute. At runtime Ant introspects on the Java class to determine the type of the attribute and converts the string in the buildfile to the proper type. For example, if the <attrType> is Boolean, Ant will convert the string "true," "yes," or "on" to the Boolean value `true`. See the Ant documents for more detail.
 - **public Object create<Element>() or public void add<Element>(<Element> o):** If the custom task allows nested elements, either a 'create' <Element> method or an 'add' <Element> method is required for each possible nested element where <Element> is the capitalized type of the nested element. If Ant encounters a nested element <Element> in the custom task while parsing the buildfile, Ant will look for the corresponding `add<Element>` or `create<Element>` in the Java class via introspection. For example, a `fileset` element can be nested in the built-in task `copy`. The Java class that implements the task `copy` defines method `public void addFileset(FileSet set)`.
- When Ant parses a project and locates a custom task, Ant will generally:
1. Create an instance of the implementation class.
 2. Invoke the 'create' <Element> or 'add' <Element> method for each nested element of type <element> when the buildfile is parsed.
 3. Invoke the 'set' <Attrname> methods for attributes named <attrname> at runtime.
 4. Invoke the 'execute' method when the task is executed.



CTIA Wireless 2002

www.ctiashow.com



To use a custom task in a project, you have two options. The first is to use taskdef element to define the custom task in the scope of the project. This element has two attributes: the name of the task and the implementation class for the task. The second option is to add the name/value pair of the task name and the class name in the 'defaults.properties' file located in the \$ANT_HOME/lib/ant.jar. This option makes the custom task available to all projects. In either case the Java implementation class needs to be included in the classpath at runtime. The following shows an example taskdef element and a sample entry in the 'defaults.properties' file. The taskdef element defines a custom task 'StopWSAppServer' in the scope of a project:

```
<taskdef name="StopWSAppServer"           classname=
"org.apache.tools.ant.taskdefs.optional.ejb.StopWSAppServer"/>
```

The following entry in the 'defaults.properties' file defines the task 'StopWSAppServer' for all projects.

```
stopWSAppServer=org.apache.tools.ant.taskdefs.optional.ejb.StopWSAppServer
```

Listing 2 is an example of a custom Ant task named stopWSAppServer. The task stops a specified application server in WebSphere Application Server AE 4.0. The stopWSAppServer task requires two attributes: nodeName and appServerName. The methods setNodeName and setAppServerName are invoked by Ant to set the two attributes. The execute method is invoked when the task is executed.

Writing Custom Ant Build Listeners

Events can be generated by Ant at certain points when processing buildfiles. Custom listeners can be created and attached to an Ant build process to receive and process these events. Custom build listeners can create custom logs, send e-mails when the build has problems or when it's complete, or integrate Ant with a GUI or IDE.

The events generated by Ant are org.apache.tools.ant.BuildEvent objects. The BuildEvent object provides access to information about the project and its components via the following methods:

```
java.lang.Throwable getException()
java.lang.String getMessage()
int getPriority()
Project getProject()
Target getTarget()
Task getTask()
```

To create a custom Ant listener, the following listener interface must be implemented: org.apache.tools.ant.BuildListener. The BuildListener interface defines the following methods:

```
public void buildFinished(BuildEvent e)
public void buildStarted(BuildEvent e)
public void messageLogged(BuildEvent e)
public void targetFinished(BuildEvent e)
public void targetStarted(BuildEvent e)
public void taskFinished(BuildEvent e)
public void taskStarted(BuildEvent e)
```

Custom buildListeners are attached to an Ant process via two methods: the addBuildListener method on a Project object and via the command line. The following is an example of how to attach a BuildListener to the Ant process via the command line with the "listener" option.

```
ant -listener org.apache.tools.ant.XmlLogger
```

The XMLLogger is a listener installed with Ant that outputs log messages in XML format. The default listener displays the messages to the standard output.

Listing 3 is an example of a simple custom Ant BuildListener. In this example the listener displays messages during the build to the standard output and sends an e-mail when the build has completed.

• • •

In conclusion, Ant is under continuous development. The next major release, Ant 2.0, will continue to improve on the three main characteristics of Ant: simplicity, understandability, and extensibility. It will provide better GUI support and integration with IDE tools. All tasks will be documented in XML files using a common DTD, which will make the tasks easier to understand. Ant 2.0 will make writing custom tasks easier by defining a clear contract between custom tasks and Ant, providing support for user-defined data types, and more. ☛

AUTHOR BIO

Ann Black is currently a WebSphere developer for IBM. She has been focusing on J2EE technology and supporting software for the last year. Ann has a bachelor's degree in electrical engineering and is a Sun-certified programmer for the Java 2 Platform.

Yu Tang, a WebSphere developer at IBM, is experienced with Java, CORBA, and J2EE. He holds a bachelor's degree in physics and a master's degree in computer science.

ANNBLACK @ US.IBM.COM

YUTANG @ US.IBM.COM

LISTING 1 Example buildfile: Build.xml

```
<project
  name="CMP11" default="do_all" basedir=".">

  <!-- set up project properties -->
  <property name="src_Dir" value="."/>
  <property
    name="bld_Dir" value="${src_Dir}/build"/>
  <property name="jar_Dir" value="${src_Dir}/jars"/>
  <property name="appName" value="cmp11"/>
  <property name="appServer" value="Default Server"/>
  <property name="nodeName" value="IBM-20MCG4EIXUG"/>
  <property
    name="hdDir"
    value="${WAS_HOME}/installedApps/CmpApp.ear"/>
  <property environment="system"/>

  <target
    name="do_all"
    depends="cleanUp, init, preDeploy_ejb, ejbdeploy, hotDeploy"/>

  <!-- Remove the build directory left from old build -->
  <target name="cleanUp">
    <delete dir="${bld_Dir}"/>
    <delete dir="${jar_Dir}"/>
  </target>
```

```
</target>

<!-- Create the build directory structure used by compile -->
<!-- and copy the deployment descriptors into it-->
<target name="init">
  <mkdir dir="${bld_Dir}"/>
  <mkdir dir="${bld_Dir}/META-INF"/>
  <mkdir dir="${jar_Dir}"/>
  <copy todir="${bld_Dir}/META-INF">
    <fileset
      dir="${src_Dir}/META-INF"></fileset>
  </copy>
</target>

<!-- Prepares source files for deployment: -->
<!-- compiles ejb source into build directory, -->
<!-- jars the compiled classes -->
<target name="preDeploy_ejb" depends="init">
  <javac
    srcdir="${src_Dir}"
    destdir="${bld_Dir}"
    extdirs="${system.WAS_EXT_DIRS}"/>
  <jar
    jarfile="${jar_Dir}/${appName}.jar"
    basedir="${bld_Dir}"></jar>
</target>
```

```

<!-- Generate deployed code for ejbs via ejbDeploy -->
<target
name="ejbdeploy" depends="preDeploy_ejb">
<exec
executable="ejbdeploy.bat"
os="Windows NT, Windows 2000">
<arg
line="${jar_Dir}/${appName}.jar c:\temp
${jar_Dir}/Deployed_${appName}.jar">
</arg>
</exec>
<exec
executable="ejbdeploy.sh"
os="AIX, SunOS, HP-UX, Linux">
<arg
line="${jar_Dir}/${appName}.jar /tmp
${jar_Dir}/Deployed_${appName}.jar">
</arg>
</exec>
</target>

```

```

<!-- Copy the deployed jar to the -->
<!-- WS installed app dir for hot deploy -->
<target name="hotDeploy">
<stopWSAppServer
nodeName="${nodeName}"
appServerName="${appServer}"/>
<copy
file="${jar_Dir}/Deployed_${appName}.jar"
tofile="${hddir}/${appName}.jar"/>
<startWSAppServer
nodeName="${nodeName}"
appServerName="${appServer}"/>
</target>

```

```

</project>

```

LISTING 2 Example WebSphere Application Server custom task

```

package org.apache.tools.ant.taskdefs.optional.ejb;

import org.apache.tools.ant.BuildException;
import org.apache.tools.ant.Task;
import org.apache.tools.ant.taskdefs.Execute;
import java.io.*;

// A custom task needs to extend the
// org.apache.tools.ant.Task class.
public class StopWSAppServer extends Task {
    String nodeName = "";
    String appServerName = "";

    // Setters for the two attributes:
    // nodeName and AppServerName
    public void setNodeName(String arg) {
        nodeName = arg;
    }
    public void setAppServerName(String arg) {
        appServerName = arg;
    }

    // Do the work
    public void execute() throws BuildException {
        if (nodeName.length() == 0) {
            System.out.println(
                "Missing the Node Name. "
                + "Please specify the name of the node "
                + "the application server resides on.");
            throw new BuildException(
                "StopWSAppServer failed. Missing Node Name.");
        }
        if (appServerName.length() == 0) {
            System.out.println(
                "Missing the Application Server Name. "
                + "Please specify the name of the application "
                + "server you want to stop.");
            throw new BuildException(
                "StopWSAppServer failed. "
                + "Missing Application Server Name.");
        }

        StringBuffer appServer =
            new StringBuffer("ApplicationServer stop {/Node:");
        appServer.append(nodeName);
        appServer.append("/ApplicationServer:");
        appServer.append(appServerName);
        appServer.append("/}");

        String[] command = new String[3];

```

```

String osName = System.getProperty("os.name");
if (osName.indexOf("Windows") >= 0)
    command[0] = "wscp.bat";
else
    command[0] = "wscp.sh";
command[1] = "-c";
command[2] = appServer.toString();

```

```

try {
    Execute.runCommand(this, command);
} catch (Exception e) {
    throw new BuildException(
        "StopWSAppServer task failed. "
        + "Could not execute command: "
        + command[0]
        + " -c "
        + command[2]
        + e);
}
}
}

```

LISTING 3 Example Ant BuildListener

```

package com.ibm.ant.example.buildlistener;

import org.apache.tools.ant.*;
import java.io.*;

// A custom listener needs to implement the
// org.apache.tools.ant.BuildListener interface
public class MailSenderBL
    implements org.apache.tools.ant.BuildListener {

    public MailSenderBL() {
    }

    public void buildStarted(BuildEvent e) {
        System.out.println(
            "Build Started: " + e.getProject().getName());
    }

    public void buildFinished(BuildEvent e) {
        System.out.println(
            "Build Finished: " + e.getProject().getName());
        System.out.println("Sending mail ....");

        // Try to send a mail message
        try {
            org.apache.tools.mail.MailMessage msg =
                new org.apache.tools.mail.MailMessage(
                    "<INSERT STMP SERVER NAME HERE>");
            msg.from("build@mycompany.com");
            msg.to("sarah@mycompany.com");
            msg.setSubject("Ant Build Complete");
            PrintStream out = msg.getPrintStream();
            out.println("Build is Complete.");
            msg.sendAndClose();
        } catch (Exception ex) {
            System.out.println(
                "Could not mail message ..." + ex.toString());
        }
    }

    public void targetStarted(BuildEvent e) {
        System.out.println(
            "Target Started: " + e.getTarget().getName());
    }

    public void targetFinished(BuildEvent e) {
        System.out.println(
            "Target Finished: " + e.getTarget().getName());
    }

    public void taskStarted(BuildEvent e) {
        System.out.println(
            "Task Started: " + e.getTask().getTaskName());
    }

    public void taskFinished(BuildEvent e) {
        System.out.println(
            "Task Finished: " + e.getTask().getTaskName());
    }

    public void messageLogged(BuildEvent e) {
    }
}

```


WebServices
JOURNAL

XML JOURNAL

THE FIRST & ONLY
WEB SERVICES
RESOURCE CD

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN

400
EXCLUSIVE

WEB SERVICES
& XML
ARTICLES



EDITED BY
SEAN RHODY

\$119
CD
VALUE

FROM
WEB SERVICES
JOURNAL

Web services **EDGE** \$100
conference & expo coupon inside

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

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

"The Secrets of the Web Services Masters"

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

Easy-to-navigate HTML format!

Bonus:

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

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

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

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

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

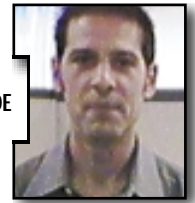
3
YEARS
25
ISSUES
400
ARTICLES
ONE CD



Now
Shipping

\$79

ONLINE
ORDER AT
JDJSTORE.COM
SAVE
\$40



Help in speeding up the building and debugging process

A Client for Testing Server-Side XML Applications

Almost every industry today has B2B exchanges that enable businesses to improve the efficiency of some process or communication with customers, partners, or suppliers. The open and public exchanges that exist today will be replaced tomorrow with Web services, which allow direct communication between two businesses.

What these two communication methods have in common is that both receive and reply using XML. If you aren't writing these communication-enabling types of applications today, you very well may be in the future.

This article demonstrates how to:

- Write a reusable tool for testing these types of applications.
- Authenticate with password-protected servers.
- Use the JDOM API.
- Extend the tool to make it more useful.

Overview

One of the problems you run into when you're writing communicative applications like these is that you need to know that you're sending and receiving proper data. Most of the projects I've worked on recently are just these types of applications – and I quickly found that I needed to determine two things: (1) the validity of the data being returned, and (2) whether the server was handling the incoming data properly.

The first – determining that the server is returning proper data – is very simple in its purest form: a request can be made via the GET method and most current browsers will render the XML that the server responds with. Although testing the validity of the response data is easy using a GET request, it is more important to know that it is responding that way as a result of being invoked with an actual request containing XML.

Which leads me to the next point, testing to make sure your server application is handling the incoming XML doc-

uments properly. What this really means is that you want to know that your application will handle an incoming XML document, and that it will perform properly. By invoking your application with XML, you can test whether, given good input, your application will respond properly and, given bad data, whether it will behave in the manner you anticipate.

I knew this wouldn't be the last application to require my testing the sending

and receiving of XML, so I decided to write a generic tool to facilitate the process.

The Tool

The tool itself is simple, and I've oversimplified it by putting almost all of the client code in one class. First let's have a look at the user interface, shown in Figure 1.

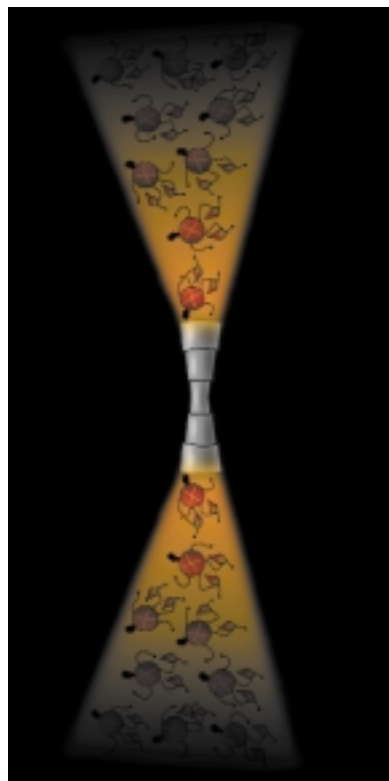
As you can tell, it's a very simple application – a button to allow you to choose an XML document, a place to type in a URL, a text box to display the results from the server, and a button to initiate the POST. A look at the code in Listing 1 gives you the following methods:

```
private void getXMLFile() & private void  
postData()
```

The method `getXMLFile` is called as a result of clicking on the XML File button. The method's only purpose is to pop open a `FileChooser` Dialog (see Figure 2) that allows you to specify the XML document you want to use for your test.

The `postData` method is what actually does the most work – too much work for an individual method, but, for simplicity, all the really important code is in one place. We start by getting an instance of a `DOMBuilder` and calling the `build()` method, passing in the File object for our XML file. We then make a connection to the server identified in the URL text field of the user interface. This is accomplished with the `URLConnection` class in the `java.net` package.

After establishing our connection, we get to use a nice class provided by the JDOM API, the `XMLOutputter` class! First we instantiate an `OutputStreamWriter`, passing the `URLConnection`'s `getOutputStream()` as a parameter to the



AUTHOR BIO

Jon Strande, president of the Harrisburg Area Java Users Group, is a consultant with Perfect Order in Mechanicsburg, PA. Perfect Order is a regional systems integrator focusing on the design, implementation, and maintenance of mission-critical computing environments.

constructor of the `OutputStreamWriter`. We then create an instance of the `XML-Outputter` class with a string representing the indent we want to use, in this case a “\t” for tabs and a boolean indicating that we’d like to use new line characters.

Now that we have that stuff set up, we need to associate our XML document object with our outputter. The method to do this is `output(Document theDoc-YouWantToWriteOut, OutputStream theActualOutputStream)`. After calling the `output()` method and a quick call to `flush()`, we’re ready to start getting input back from the server.

The code for this is nothing new, so I’ll only touch on it by saying that we’re getting the input stream from `URLConnection` and that you can examine all this code by downloading the source.

What we’ve accomplished here is an easy way to post an XML file to a server and display the results in a text box. For testing purposes I’ve written a servlet that pretty much echoes the content of

the XML document submitted by the client. The servlet adds a node to the document named `SERVER_TIME`, which contains a date-time stamp as the text of that node.

The code for the `doPost()` method is shown in Listing 2. The code for the servlet is simple and can also be examined in Listing 2 or by downloading the source code from this article.

Authenticating the Use of the Authenticator Class

One problem with these applications is that they’re generally password protected – in order to access them you need to be authenticated. Thankfully, Java makes it easy to do this. In the `java.net` package is an `Authenticator` abstract class that you can subclass to create a graphical interface that allows users to provide their credentials for authentication. The code to do this same thing can be found in Chapter 7, “Retrieving Data with URLs,” of *Java Network Programming*, 2nd Edition, by Elliotte Rusty Harold (O’Reilly).

A simple overview of this class, right from the `JavaDoc`, is as follows:

The class `Authenticator` represents an object that knows how to obtain authentication for a network connection. Usually, it will do this by prompting the user for information.

Applications use this class by creating a subclass, and registering an instance of that subclass with the system with `setDefault()`. When authentication is required, the system will invoke a method on the subclass (like `getPasswordAuthentication()`). The subclass’s method can query about the authentication being requested with a number of inherited methods (`getRequestingXXX()`), and form an appropriate message for the user.

So, as you can see by the code below, we have the following constructor and methods:

```
private void show()
public PasswordAuthentication
getPasswordAuthentication()
```

Those methods, as well as two inner classes that handle the OK and Cancel responses (again, right from *Java Network Programming*), are all that’s really required to add authentication to this application. The `Authenticator` will prompt you for credentials only when the site you’re trying to access requires it, as shown in Figure 3.

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



FIGURE 1 Screenshot of the main application

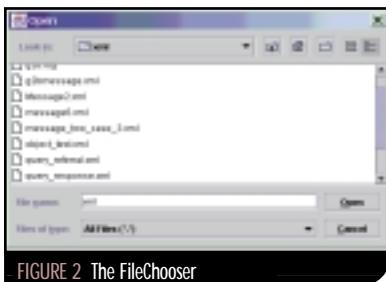


FIGURE 2 The FileChooser

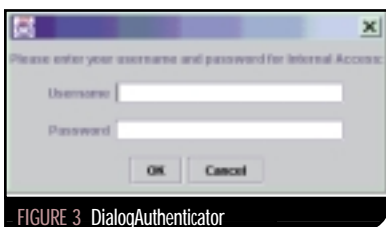


FIGURE 3 DialogAuthenticator

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 March *JDJ*:

Extending the J2SE 1.4 Logging Package
Monitoring made easy

Jini Surrogate as a Platform for J2ME Games
Surrogate architecture incorporates smaller devices

Integrating J2ME, GPS, and the Wireless Web
Developing location-based applications

Book Review:
Java Internationalization

Java and Wireless
Building end-to-end Palm applications using Java

Mac OS X & Java
A perfect marriage



Extending the Tool

As I mentioned, the tool itself is very basic, and is useful for testing XML-based applications. And there are quite a few ways that you can turn this application into a more effective testing tool. For example:

- You could change the file chooser to point to a directory and load all the XML documents in that directory to be posted to the server.
- Instead of loading static XML documents from a directory, you could write a Message Creator type class that builds XML messages with real data. Doing this doesn't prevent you from reusing the tool with other applications. All you have to do is write a Message Creatable interface and implement a new Message Creator for every new application you need to test.
- You could use threads to make simultaneous posts to the server to test real-life loads. This could be combined with the previous suggestion to really test how your application will handle incoming requests.

The JDOM API

This tool was written using the JDOM (www.jdom.org) beta 6. However, none of the methods used in this application were deprecated in the beta 7 release of the JDOM API. That said, let's look at some of the classes and methods being used.

For starters, we're using the org.jdom.Document class, which defines

the methods for manipulating an XML document. I make more use of this class in the sample servlet that I've included. The method I'm using is getRootElement().

The next class I use, the Element class, is analogous to a node in an XML document or an "element" in the document. The method addContent() is used in the Element class to add text to an element; those elements can then be added to other elements through the addContent() method, or to a document object.

As already mentioned, I'm using two other classes, a DOMBuilder and an XMLOutputter. Together these classes allow you to (1) create a Document object from various types of inputs and (2) write a Document object out, respectively.

Overall, the JDOM API is extremely easy to use; I may have made it seem more difficult than it truly is. The authors of JDOM have done a great job of creating a simple and effective API for working with XML.

Conclusion

As you can see, testing an XML-based server-side application is fairly straightforward and can be accomplished rather easily. The basic tool, combined with one or more of the suggestions for extending the tool, should help speed up the process of building and debugging server-based XML applications. ☒

JSTRANDE @ POSS.COM

LISTING 1 private void getXMLFile() & private void postData()

```
private void getXMLFile()
{
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
    int result = fileChooser.showOpenDialog(this);
    if(result == JFileChooser.CANCEL_OPTION)
        xmlFile = null;
    else
        xmlFile = fileChooser.getSelectedFile();
}

private void postData()
{
    try
    {
        Document document = null;
        DOMBuilder builder = new DOMBuilder(false);
        document = builder.build(xmlFile);
        if(document != null)
        {
            postURL = urlField.getText();
            URL url = new URL(postURL);
            System.out.println("after new URL(...)");
            HttpURLConnection urlCon =
                (HttpURLConnection)url.openConnection();
            urlCon.setDoOutput(true);
            OutputStreamWriter out = new
                OutputStreamWriter(urlCon.getOutputStream());
            XMLOutputter outputter = new XMLOutputter("\t",true);
```

```

        outputter.output(document, out);
        out.flush();
        StringBuffer sb = new StringBuffer();
        InputStream in = urlCon.getInputStream();
        BufferedReader br = new BufferedReader
            (new InputStreamReader(in));
        String lineOfText;
        try
        {
            while((lineOfText = br.readLine()) != null)
            {
                sb.append(lineOfText);
            }
        }
        catch(Exception e)
        {
            textArea.append(e.toString());
        }
        textArea.append(sb.toString());
    }
}
catch(Exception e)
{
    System.out.println(e.toString());
}
}

```

LISTING 2 doPost method in servlet

```

public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException
{
    PrintWriter out = new PrintWriter
        (response.getOutputStream());
    DOMBuilder builder = new DOMBuilder(false);
    Document outgoingDocument = null;
    try
    {
        InputStream inputStream = (java.io.InputStream)
            request.getInputStream();
        try
        {
            /*
                replace this code with code that calls your
                application
                specific code, perhaps an app controller.
                outgoingDocument =
                messageController.handleMessage(inputStream);
            */
            Document document = builder.build(inputStream);
            Element root = document.getRootElement();
            Element newRoot = root;
            Element element = new Element("SERVER_TIME");
            element.addContent(new java.util.Date().toString());
            newRoot.addContent(element);
            outgoingDocument = new Document(newRoot);
        }
        catch(org.jdom.JDOMException jdomException)
        {
            outgoingDocument = buildErrorXMLDocument
                (jdomException.getMessage());
        }
    }
    catch(Exception e)
    {
        outgoingDocument = buildErrorXMLDocument(e.getMessage());
    }
    finally
    {
        response.setContentType("text/xml; charset=UTF-8" );
        XMLOutputter outputter = new XMLOutputter("\t",true);
        outputter.output(outgoingDocument, out);
        out.flush();
        out.close();
    }
}

```

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

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!

Wireless Business & Technology:

Wireless Checks In to Hospitality

Wireless is transforming the hotel experience for guests and employees alike.

Is Worldwide Wireless Broadband Barrelling Our Way?

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

Making Money from Messages

SMS might be the solution to the growing debt problem faced by companies that massively overbid for 3G licenses.

Mobile Operators Seek Incremental Revenue from the Mobile Internet

Leveraging infrastructure for ROI.





Stylesheets separate content from presentation

Transforming XML Documents into HTML

The power and elegance of XSLT – the Extensible Stylesheet Language for Transformations – stems from its ability to transform XML documents into other output formats like HTML, fulfilling one of the original promises of XML: separating content from presentation.

XSLT is particularly powerful because a single stylesheet can format all the XML documents conforming to a DTD into HTML for publication on a Web site. The stylesheet can also be used to automatically generate such features as a hyperlinked table of contents, the building of which requires substantial manual work without XML.

XSLT is also elegant, because if you need to reformat all the pages in your Web site, for instance, you need to change the code in only one place, the stylesheet, so long as your source documents are in XML.

Written primarily for content authors, technical writers, Web designers, and other nonprogrammers, this tutorial aims to demonstrate some of XSLT's power and elegance in separating content from presentation and in automatically generating narrative-oriented HTML documents while showing you how to create progressively more complex template rules – rules that are at the core of XSLT.

Review: The Template Rule

In its most basic form an XSLT stylesheet uses what are called *template rules* to match nodes in an XML document and transform them into another format. A template rule is an XSL element that matches a node in the XML source document and typically applies an output format, such as HTML, to it.

For example, say you have the following simple XML document:

```
<?xml version="1.0"?>
<document>
<body>You've probably heard the
propaganda by now: XML blesses you
with a way to separate content from
presentation.</body>
</document>
```

You can use a template rule to find the children of the root element and to format its contents in HTML for presentation. Here's a template rule that does just that:

```
<xsl:template match="/">
  <html>
    <body>
      <xsl:apply-templates/>
    </p>
  </body>
</html>
</xsl:template>
```

In this rule the `<xsl:template match="/">` element uses the value of its `match` attribute to find a node in the XML source. The forward slash operator, an XPath expression, specifies the document's root node. The rule could also match on the same node by specifying it explicitly in the template rule, that is, `<xsl:template match="docu-`

ment">. By matching on the root node of the document, we're able to build an HTML container that provides the skeleton code (here, just `<html>` and `<body>`) for our document. I'll expand on this concept below.

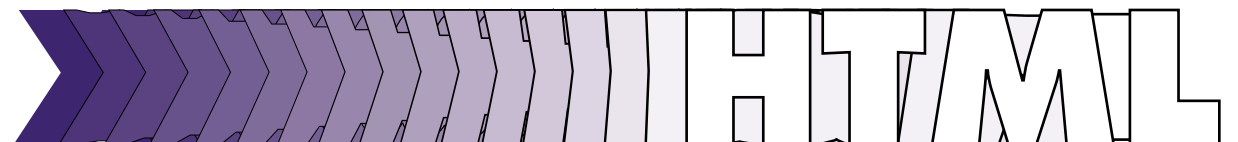
The `<xsl:apply-templates>` element invokes a built-in XSLT template rule that processes the children of the matched node, meaning roughly that it outputs the children. Because there's only one child of the `<document>` node in our XML file, `<xsl:apply-templates>` suffices to print the meager contents of the file. If, however, the `<document>` element contained more children, `<xsl:apply-templates>` would print all of them out, too, and we'd want additional template rules to control how they're formatted.

Before we can push our source XML and its accompanying stylesheet through an XSLT processor to render the HTML output, we need to do a couple more things. First, we need to add an XML processing instruction to the top of the stylesheet. Second, we must wrap the template rule with the `<xsl:stylesheet>` element, which all XSL stylesheets require as their top-level element, and set a namespace for it (note that some versions of Internet Explorer and the MSXML parser may require a different namespace; see <http://msdn.microsoft.com/> and the Unofficial MSXML XSLT FAQ at www.netcrucible.com/xslt/msxml-faq.htm for details):

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

AUTHOR BIO

Steve Hoenisch, a technical writer/consultant with Verizon Wireless, is a former journalist and teacher. He has been developing Web sites since 1996.



```

version="1.0">
<xsl:template match="/">
  <html>
    <body>
      <p><xsl:apply-
        templates/></p>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>

```

We now have a minimal stylesheet that will convert our XML source to HTML. To view the output in Internet Explorer 5.5 or later, we first need to make a change to the XML source document: it must include a stylesheet processing instruction (appended after the XML processing instruction) that references our new minimal stylesheet. In this case it's located in the same directory as the source file, as the path in the href attribute's value testifies:

```

<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href=
  "my_stylesheet.xsl"?>
<document>
<body>You've probably heard the
  propaganda by now: XML blesses you
  with a way to separate content from
  presentation.</body>
</document>

```

Let's expand our minimal stylesheet to do a few more things. First, we'll modify our template rule to output the body element specifically (as opposed to all the children of the root element); second, we'll add a link to a Cascading Style Sheet that specifies the visual properties of our HTML formatting:

```

<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform"
  version="1.0" >
  <xsl:template match="document">
    <html>
      <head>
        <link rel=
          "stylesheet" type=
            "text/css" href=
              "doc.css"/>
      </head>
      <body>
        <p><xsl:value-of
          select="body"/></p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

```

This stylesheet first matches the root node explicitly by name (document), builds an HTML container for it as before, and then uses the <xsl:value-of>

element to select and output the text contents of the message node. The select attribute identifies the element whose contents are to be processed.

Notice that in the stylesheet I also added a link to a CSS file in the same directory as the XSLT stylesheet. The CSS file contains the code in Listing 1 (all listings can be found at www.sys-con.com/xml/sourcec.cfm). In this example only a few of the CSS rules are used, but the rest of them will be put into play as we expand our rudimentary stylesheet to process a more complex XML document.

Using CSS to complement XSLT is a powerful strategy for building Web pages – a strategy that splits presentation into what I call *formatting* and *styling*. Formatting can be seen to include basic HTML markup like headings, horizontal rules, lists, and the like. Styling, meantime, defines the visual properties of markup: its colors, sizes, widths, margins, bullet types, and so forth. Separating visual styling from formatting gives you a way to make wholesale design changes to a Web site without having to change the formatting code in every HTML document; if you've set up your Web pages properly, with all of them linking to a single CSS, you merely make the stylistic changes in one file, the Cascading Style Sheet.

Building a Complex Stylesheet

This section builds on the review above to create an XSLT stylesheet that will transform any document that conforms to our DTD into an HTML document. We'll create the stylesheet from the top down, step by step, explicating most of the template rules as we go.

Consider the generic XML document in Listing 2.

The document is based on the DTD in Listing 3. (This DTD is very loosely based on a scaled-down version of the TEI Lite DTD with some XHTML and other customizations thrown in; it's been constructed to demonstrate the XSL transformations in this tutorial and, unlike TEI Lite, isn't suitable for other uses. For information about TEI Lite see www.tei-c.org/.)

The XSLT stylesheet in Listing 4 is also based on the DTD. Why bring a DTD into this discussion? The answer is that it's best to build the stylesheet based on your DTD to ensure that all elements and attributes are processed fully and according to your requirements. The DTD provides the cues by which you build your stylesheet. That is, to build a suitable stylesheet – one that processes all the elements and attri-

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 March **WSJ**:

Making Second-Generation Web Services Secure

First-generation Web services will use existing technologies...but what's next?

Securing Web Services

Security is vital if Web services are ever to achieve mass deployment

Scalable Web Services Using JMS and JCACHE

A solution to the problems inherent in delivering large-scale distributed systems

Web Services Impact on Business Process Management

The promise of a single solution for integration across multiple enterprises



“Separating visual styling from formatting gives you a way to make wholesale design changes to a Web site without having to change the formatting code in every HTML document”

butes in your XML document fully and appropriately – you should develop your stylesheet based on your DTD, not on an XML document alone. For more information about analyzing DTDs to develop stylesheets, see Chapter 21, “DTD Analysis,” in *The XSL Companion*, by Neil Bradley (Addison-Wesley).

Let's start building the stylesheet. Since XSL stylesheets are themselves XML documents, they should generally begin with an XML processing instruction as their first line: `<?xml version="1.0"?>`. The next line contains an optional document type declaration for the stylesheet to specify the stylesheet's root element, `xsl:stylesheet`. I'm including the document type declaration in our stylesheet because I want to declare several general entities as an internal DTD subset for use in the stylesheet. (Recall that general entities let you replace an entity with its value; that is, wherever the entity `pub.date` appears in the stylesheet as text, it is replaced by its value as defined in the entity declaration, here February 8, 2002.)

```
<!DOCTYPE xsl:stylesheet [
  <!ENTITY pub.date "February 8, 2002">
  <!ENTITY mdash "--">
  <!ENTITY nbsp "&#160;">
]>
```

The root element for an XSLT stylesheet must be either `xsl:stylesheet` or `xsl:transform`. The `xsl:` prefix before stylesheet and transform is the customary XSL namespace; all the XSL elements are in the www.w3.org/1999/XSL/Transform namespace, which must be referenced as the value of the `xmlns:xsl` attribute of the `xsl:stylesheet` root element, as I've done here (see the chapter titled “XSL Transformations” in O'Reilly's *XML in a Nutshell* for additional information about the namespace):

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

Next, I set the output method to HTML and include a reference to the HTML version I want to use:

```
<xsl:output method=
  "html" doctype-public="--/W3C//DTD
  HTML 4.0 Transitional//EN"/>
```

Creating an HTML Container

Now, bringing a simple template-matching rule into play, we begin building the HTML skeleton for the output file:

```
<xsl:template match=
  "document">[...HTML SKELETON
  CODE...]</xsl:template>
```

This template rule selects the XML document's root element, `<document>`, and circumfuses it with HTML code, forming a skeleton that will contain the body of the XML document. In the case of this stylesheet, the skeleton is in the form of a header, a content area, and a footer.

In the header I've used an XSL element – `<xsl:value-of select="docinfo/title"/>` – to select the title of the XML document and to output it in the title slot of the HTML document that I'm building.

Notice, too, that between the HTML `<head>` tags I've also created an HTML link to a Cascading Style Sheet, which contains as much of the styling information for my HTML formatting as possible:

```
<link rel="stylesheet" type=
  "text/css" href="doc.css"/>
```

In the next XSL command I again select the value of the `<title>` tag in the XML document, this time for use as a headline atop the HTML document:

```
<xsl:value-of select=
  "docinfo/title"/>
```

Skip over the rest of the HTML code in the header now and bore into the content area until you find the next command prefaced by the XSL namespace:

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

This template rule selects the entire content of the body element in the XML document, processes it according to the other template rules in the stylesheet, and outputs the result at the point of this command inside the HTML container that we built when we matched on the root element earlier.

Formatting the Body

To format the body of the XML document, my stylesheet includes a template rule that selects the body element: `<xsl:template match="body">`. This template does three things:

1. It inserts a Table of Contents headline.
2. It calls another template that dynamically builds a table of contents.
3. It processes all `<div1>` elements.

The `<xsl:call-template name="toc"/>` command calls a macro (located toward the end of the stylesheet in the Macros section) that cycles through the six levels of `<div>` elements, makes a reference to the `<div>` element's heading in the form of a hyperlink, and numbers the section headings using the 1.1 format. (You can see what the output looks like by downloading the XML source document from the *XML-Journal* Web site and opening it in Internet Explorer 5.5; the hyperlinks in the displayed document are brown but aren't underlined.)

The set of macros that build the table of contents originated in the W3C's XSLT stylesheet for the XML Recommendation.

“Reverse engineering it is a potent method of teaching yourself some of the advanced capabilities of XSLT”

Reverse engineering it is a potent method of teaching yourself some of the advanced capabilities of XSLT. The W3C's stylesheet is available in the download file for Chapter 9 of Michael Kay's book, *XSLT Programmer's Reference*, at www.wrox.com/. In IE5.0 or greater you can view the XML source code for the W3C's XML Recommendation at www.w3.org/TR/2000/REC-xml-20001006.xml or you can download it and view it using an editor like XML Spy.

The next set of templates in our stylesheet creates headings for the <div> elements, mapping the <head> element in <div1> to the HTML heading element <h1> and so forth through <div6> and <h6>. The template-matching rules for the head elements also call another template, named "head", which inserts an ID attribute in each heading and numbers them with the command mode="number".

Processing the Child Nodes

Finally, after building the table of contents and the headings for each <div> section, the stylesheet uses the following code to process all the children of the <div1> element:

```
<xsl:template match="div1">
  <xsl:apply-templates/>
</xsl:template>
```

In this code the <xsl:apply-templates/> simply tells the XSLT processor to process all the child elements of <div1> and to output them according to the template rules for each child element listed further down in the stylesheet. Many of these child elements are processed simply by what I've called the *XHTML Quick Print Module*. This template rule selects any of the elements listed in the value of the match attribute, makes a copy of the element, makes a copy of all the element's attributes, and passes them through to the target output file intact. Because my DTD contains several XHTML elements for low-lying nodes, I can use this template rule instead of writing a separate one for each XHTML element.

The complete stylesheet is presented in Listing 4.

Further Reading

On the heels of this column's terse introduction to XSLT, I suggest you spend a couple hours in your neighborhood bookstore reading the following chapters in this order:

- Chapter 6, "Transformation: Repurposing Documents," in O'Reilly's *Learning XML*
- Chapter 8, "XSL Transformations," in O'Reilly's *XML in a Nutshell*

- Whatever selections you deem useful from Wrox's *XSLT Programmer's Reference* by Michael Kay

Finally, take some time to browse through the resources listed on the World Wide Web Consortium's XSL page at www.w3.org/Style/XSL/. The page includes links to other tutorials, a FAQ, the XSL specifications, and a variety of resources.

I also suggest you start playing around with XSLT a little on your own; it's the best way to learn how to use it. Try playing with the code accompanying this column: add some elements to the DTD and the XML document and then write template-matching rules to process and output them in Internet Explorer. You may have to install the newest version of the MSXML parser to get the code to display correctly; see <http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/766/msdncompositedoc.xml> for details.

This column has only scratched the surface of XSLT. It's a complex but powerful programming language that, once you learn how to use it, yields vast gains in productivity that spring from the original promise of XML – separating content from presentation. ☼

SHOENISH @ RCN.COM

SAVE UP TO \$100 ON MULTIPLE SUBSCRIPTIONS



Special online offers

Pick 4 or 5 and Subscribe

for one **special low price**

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



COVER SUBJECT TO CHANGE WITHOUT NOTICE

Wireless Business & Technology • Java Developer's Journal
 Web Services Journal • WebLogic Developer's Journal
 XML-Journal • WebSphere Developer's Journal
 ColdFusion Developer's Journal • PowerBuilder Developer's Journal

WWW.SYS-CON.COM/SUBOFFER.CFM

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 cool XML technology!

XML-J Specials



Microsoft Visual Studio .NET
Enterprise Architect

Bestsellers

1. JBuilder Enterprise Upgrade 5.0 from JBuilder 5.5 Enterprise \$1799.99
2. Borland JBuilder Pro V6.5 Single \$949.99
3. JBuilder Enterprise Upgrade 5.0 from any previous Enterprise Edition \$2499.99
4. JBuilder 6 Personal 99232015/Unix/StdEd \$95.99
5. Microsoft Visual C++ 4.5.2 Enterprise Suite \$2749.99

What's Online

www.sys-con.com/xml

XML-Journal.com

Visit www.xml-journal.com for the latest industry news and events from the world's leading XML resource. Get a leg up in your own job by learning what's happening in the IT world.

Readers' Choice Awards

Known as the "Oscars of the software industry," the 2002 **XML-J Readers' Choice Awards** feature an expanded list of product categories and other additions, ensuring that this year's awards will be the best ever. Vote online for your favorite products between April 1 and June 30 and help decide the winners!

2002 Edge International East Conference

Learn about and sign up for **SYS-CON's Web Services Edge 2002 East, JDJEdge 2002 East, and XMLEdge 2002 East International Conference & Expo**, to be held at the Jacob Javits Convention Center in New York City. This four-day conference kicks off on June 24, and will feature the leading information technology professionals in the industry, who'll share their knowledge of Java, XML, and Web services. This event will coincide with the New York Technology Exchange to include the world's greatest business technology members and products.

Tune in to **SYS-CON TV** at www.sys-con.com/xml for highlights from past conferences!

Free Weekly Newsletters

SYS-CON's industry newsletters are informative, convenient, up to the minute – and absolutely **FREE**. Just look in your e-mail inbox every week. Learn what's hot and what's not at your leisure, because these newsletters can be read anywhere you have access to a computer. Subscribe to **XML-J Industry Newsletter**, and check out the other **SYS-CON**-family newsletters while you're at it for news on Java, Web services, wireless, Linux, and ColdFusion. Sign up at www.sys-con.com!

What's Your Opinion

XML-Journal Online offers you not only the opportunity to read our articles online, but to tell us what you think of them. What did you like? What made you mad? What would you like to see in the magazine? You can even e-mail our authors directly with your questions.

Note: Some feedback may be selected for our print edition.

XML Forums

Join **XMLJ List**, a free XML mailing list community, and discuss the most important IT issues with industry professionals and **XML-Journal** writers. Voice your opinion...ask questions...read what others have to say.

Come to Our Archives

Did the dog eat your first issue of **XML-J**? Did you throw out last July by mistake? All is not lost. Rediscover the past by going to the complete **XML-J** archives. Access is **FREE** with your paid subscription. Become the most informed IT professional in the business by reading www.xml-journal.com.



QUICK POLL

Are you working with Web Services?
A- Yes
B- No

Vote
Results



XML-J ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com		64
BEA	www.bea.com/download		9
CTIA Wireless 2002	www.ctishow.com		43
HitSoftware	www.hitsw.com.com	800-345-4001	13
Java Developer's Journal	www.sys-con.com	800-513-7111	50
JavaOne	http://java.sun.com/javaone	888-886-8769	37
JDJ EDGE Conference & Expo	www.sys-con.com	201-802-3057	27
JDJ Store	www.jdjstore.com	888-303-JAVA	19, 58-59
Macromedia	www.macromedia.com/downloads	888-939-2545	4
.Net Developer Conference & Expo	http://events.internet.com		17
SilverStream Software, Inc.	www.silverstream.com	888-823-9700	6
Sonic Software Corporation	www.sonicsoftware.com		2, 3
SYS-CON Media	www.sys-con.com	800-513-7111	25, 55, 57
SYS-CON Reprints	www.sys-con.com	201-802-3026	35
VoiceXML Planet Conference & Expo	http://events.internet.com		17
Web Services Edge Conference & Expo	www.sys-con.com	201-802-3069	33, 38-39
Web Services Journal	www.sys-con.com	800-513-7111	53
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	800-513-7111	23
WebSphere Developer's Journal	www.sys-con.com/websphere	800-513-7111	35
Wireless Business & Technology	www.sys-con.com/wireless	800-513-7111	51
Wireless EDGE Conference & Expo	www.sys-con.com	201-802-3069	46, 47
XML EDGE Conference & Expo	www.sys-con.com	201-802-3056	29
XML Global	www.xmlglobaltechnologies.com/yourinformation	800-201-1848 ext.210	63
XML-Journal	www.sys-con.com	800-513-7111	31, 49

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



www.wbt2.com

SUBSCRIBE NOW

www.javadevelopersjournal.com

TO THE

www.sys-con.com/xml

FINEST

www.coldfusionjournal.com

TECHNICAL

www.sys-con.com/pbdj

JOURNALS

www.webspheredevlopersjournal.com

IN THE

www.wldj.com

INDUSTRY!

www.wsj2.com



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

SYS-CON MEDIA

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

Deploying Web Services on WebSphere

Ron Ben-Natan discusses the tools and services provided by WebSphere through a real example - from the conceptualizing of the service as a business function that's useful for remote and decoupled usage, through the packaging of the implementation as an enterprise application and all the way to enabling it through Apache SOAP, creating the WSDL, and publishing it in a UDDI registry.

XML Messaging Part 2

Sending messages asynchronously and using the JAXM tag library

Mike Jasnowski shows how to use the custom tag library that's distributed with the JAXM API to incorporate sending SOAP messages using Java Server Pages.

A Pragmatic Convergence of ebXML & Web Services

John Aloisius Ogilvie explains how ebXML has done the industry a disservice by promising a set of comprehensive standards but delivering a set of academic working papers or simple rubber-stamping of more useful vendor-sponsored standards.

Next Month...



XML JOURNAL

Don't miss the April issue!

JDJSTORE.COM GUARANTEED! LOWEST PRICES!

Guaranteed Best Prices

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

Terms of offer:

- Offer good through May 31, 2002
- Only applicable to pricing on current versions of software
- Offer does not apply toward errors in competitors' printed prices
- Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

Attention Software Vendors:

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

BORLAND JBuilder Enterprise V6.0

FREE with order...Your choice of one of "The Complete Works" CDs



JDJStore.com **\$2749⁹⁹**



SYS-CON MEDIA
JDJ, CFDJ, XML-J, or CFAdvisor
The Complete Works

WEBGAIN Visual Café 4.5.2 Enterprise Suite

FREE with order...Your choice of one of "The Complete Works" CDs



JDJStore.com **\$2749⁹⁹**

MACROMEDIA

ColdFusion Server 5 Professional Edition

ColdFusion Server 5 Professional Edition
Macromedia® ColdFusion® 5 empowers Web developers with a highly productive solution for building and delivering the next generation of Web applications. ColdFusion 5 offers major enhancements in development, administration, and performance.



ColdFusion Server 5 **\$1219⁹⁹**

INSTALLSHIELD

InstallShield Professional 6.3

InstallShield Professional 6.3 is the de-facto industry standard for creating sophisticated Windows installations. InstallShield Professional 6.3 offers scripting and visual tools in an integrated environment that gives users complete flexibility and control while allowing unparalleled productivity.



InstallShield Professional 6.3 **\$999⁰⁰**

SYS-CON MEDIA

XML-Journal: The Complete Library

THE MOST COMPLETE LIBRARY OF EXCLUSIVE XML-J ARTICLES IN ONE CD
Features & Articles, Product Reviews
Case Studies, Tips & Tricks, Interviews,
Editorials, IMHOs & more!



XML-Journal: The Complete Library **\$53⁹⁹**

INSTALLSHIELD

InstallShield Professional 6.3 Upgrade from 6.X

InstallShield Professional 6.3 is the de-facto industry standard for creating sophisticated Windows installations. InstallShield Professional 6.3 offers scripting and visual tools in an integrated environment that gives users complete flexibility and control while allowing unparalleled productivity.



InstallShield Professional 6.3 Upgrade from 6.X **\$499⁰⁰**

BORLAND

Borland Delphi 6 Enterprise

Delphi 6 makes next-generation e-business 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 **\$2849⁹⁹**

eHELP

RoboHELP Office 2002

RoboHELP Office 2002 from eHelp Develop Help for your Java Apps with RoboHELP Office 2002. Just point and click to develop the Help content, structure and design no coding. Easily deploy these Help formats:

- WebHelp - Web, server and desktop-based cross-browser, cross-platform Help
- Oracle Help for Java 100% pure Java solution, not Oracle dependent
- JavaHelp - 100% pure Java solution



RoboHELP Office 2002 **\$898⁹⁹**

ORDER TODAY!

888-303-JAVA

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



**GUARANTEED
BEST PRICES
FOR ALL YOUR
SOFTWARE AND
HARDWARE
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**

ALTOWEB

Application Platform Release 2.5

The AltoWeb Application Platform lets you build, deploy, and manage J2EE applications and Web Services up to 10x faster without requiring extensive J2EE or Web Services expertise. How? By replacing lengthy, custom and complex J2EE, XML and Web Services coding with rapid component assembly and reuse.



Application Platform Release 2.5 **\$3540⁰⁰**

ALTOVA

XML Spy v 4.2 Suite

Countless XML technologies - One powerful XML Spy 4.2 Suite! XML Spy now offers a comprehensive and easy-to-use product family to facilitate all aspects of Advanced XML Application Development.



XML Spy v 4.2 Suite **\$399⁹⁹**

HP

Jornada 547 Handheld Device

As a busy professional, you know every day can be a challenge. Meetings, appointments, projects, and deadlines sometimes it's hard keeping it all together. But now there's something that can help you handle it all with ease...the HP Jornada 547 Color Pocket PC. It has everything you need to gracefully juggle the details...and have fun while you're doing it.



Jornada 547 handheld Device. **\$299⁰⁰**

SITRAKA

JClass Enterprise Suite v 5.0 Bytecode/Gold Support

Fully scalable to mission-critical development environments, this award-winning suite provides a wide range of high-value GUI functionality including: Tables and grids charting and graphing layout and reporting hierarchical data display data connectivity GUI enhancements data input and validation JAR optimization Everything the professional Java developer needs to build powerful and flexible application front ends is here. Plus, JClass components offer unrivaled IDE, JDK and platform support, and include one year of premium technical support and free upgrades.



JClass Enterprise Suite v 5.0 **\$2945⁰⁰**

METROWERKS

CodeWarrior Pro V6.0

CodeWarrior for Java® is a powerful, award-winning integrated development environment for programming 100% pure Java applications from desktop to wireless devices using Sun's J2SE, PersonalJava, and J2ME standards. CodeWarrior's open and customizable environment provides developers with productivity, support for team collaboration, and fastest time to market for creating innovative Java applications and services.



CodeWarrior Pro V6.0 **\$349⁰⁰**

SEIKO

SmartPad 2

Seiko Instruments SmartPad2® The Seiko Instruments SmartPad2 is the first product that lets you instantly capture everything you write or draw using the SmartPad pen on ordinary paper. You can hand-write notes and store them in your Date Book or draw a map with directions and attach it to a contact in your Address Book.



SmartPad SP-580 Notepad for Palm Organizer . . . **\$198⁹⁹**

INSTALLSHIELD

DemoShield 7.0 Full

DemoShield has the powerful and easy-to-use features that users need to create interactive, multimedia software demos for Web demonstrations, Computer-based Training, CD catalogs, and more. Showcase your product, introduce your application, launch your installation - create an interactive dynamic demo that will increase the value of your product or service with DemoShield.



DemoShield 7.0 Full **\$599⁰⁰**

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**

WEBGAIN

VisualCafé Expert 4.5.1

WebGain VisualCafé 4.5 Expert Edition is now bundled with Macromedia® Dreamweaver® UltraDev® 4.0 and Turbo XML to create a powerful JSP, XML, Java development solution for professional developers. Build pure Java applications with the VisualCafé intuitive 2-way visual development environment, powerful debugger and faster compilers.



WebGain VisualCafé Expert Edition **\$899⁰⁰**

MACROMEDIA®

JRun™ Server 3.0/3.1 Enterprise 1 CPU Licenses

JRun™ 3.0/3.1 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 3.0/3.1 Enterprise (1 CPU Licenses) **\$4499⁰⁰**

SITRAKA

JClass Chart 5.0 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 5.0 Bytecode/ Gold Support **\$1234⁰⁰**

XML NEWS

New Series of E-Learning Courses

(Ottawa) – Online-learning.com has a new series of advanced online XML courses for technical writers, information designers, and Web professionals. The courses have been developed in collaboration with Ken Holman, a leading XML expert.

ONLINE-LEARNING.com

"Practical Guide to XSLT," the first in the series, features 40 hours of learning material, nine hands-on assignments, and an electronic copy of Holman's book, *Practical Transformation Using XSLT and XPath*. Two additional courses, "Practical Guide to XSLFO" and "Practical Guide to XML Modeling," are planned for later this year.

www.online-learning.com

Book on Jabber Protocols Available

(Sebastopol, CA) – O'Reilly is now shipping *Programming Jabber: Extending XML Messaging* by DJ Adams.

Jabber, a set of protocols expressed in XML, is an extensible framework that allows people and applications to exchange simple text messages used to extend the backbone of an enterprise data system. Jabber gives programmers the power to build applications that have identity and presence, and that can take part in conversations.

O'REILLY



OASIS to Develop Spec for Multilingual Data Exchange

(Boston) – Members of OASIS have formed the OASIS XML Localization Interchange File Format (XLIFF) Technical



Committee to advance a specification for multilingual data

exchange. XLIFF will allow any software provider to produce a single interchange format that can be delivered to and understood by any localization service provider.

Supporting the entire localization process, XLIFF will be product-independent, and open enough to allow the development of tools compatible with an implementer's own proprietary data formats and company culture.

OASIS will host an open mail list for public comment on XLIFF, and completed work will be freely available to the public without licensing or other fees.

www.oasis-open.org

InfoSpace Speech Platform, VoiceGenie Gateway to Join

(Bellevue, WA / Toronto) –

InfoSpace, Inc., a provider of wireless and Internet software and appli-



cation services, has announced an agreement with VoiceGenie Technologies Inc., one of the leading

providers of complete VoiceXML gateway solutions, to integrate VoiceGenie's VoiceXML gateway



into InfoSpace's speech platform.

The combination will provide wireless carriers with support for multiple ASR (automatic speech recognition) engines and international telephone standards, as well as the ability to deploy multimodal applications on a VoiceXML 2.0-compliant gateway.

www.infospace.com

www.voicegenie.com

Percussion Software Unveils Authoring Accelerator

(Stoneham, MA) – The newly available Rhythmyx authoring accelerator for Microsoft Word 2000 from Percussion Software provides seamless interaction between Rhythmyx and Microsoft Word, eliminating the need for third-party application development and support.



Key elements include round-tripping support, ability to filter, field extraction, in-line system links and images, and metadata.

www.percussion.com

Fiorano to Use B-Bop Tool for Tifosi Platform

(Burlingame, CA) – Fiorano Software, Inc., has selected B-Bop Xfinity Designer as a preferred

Fiorano tool for mapping disparate XML formats within Tifosi, Fiorano's next-generation enterprise integration and business process management platform. B-Bop Xfinity Designer is a visual tool for developers to quickly and easily generate XSLT stylesheets without extensive programming, speeding up time-to-market.



www.fiorano.com

www.b-bop.com

eXcelon's Native XML Database in Release Three

(Burlington, MA) – eXcelon Corporation has announced Release Three of eXtensible Information Server (XIS), its native XML database management system.

Release Three has been updated to embrace the latest XML standards: initial support for XQuery, XPath 1.0, XSLT 1.0,

including Java extensions (in anticipation of the XSLT 2.0 standard), and XML Schema 1.0.

It also includes support for HP-UX (11.0 and 11i), fully automatic query optimization into the XPath engine, transaction "batching," full support for JDK 1.3, and additional performance-tuning options to increase overall throughput, enhanced user management, and user authentication.

www.exceloncorp.com

IONA Platform Offers ebXML Support

(Waltham, MA) – IONA has announced the general availability of the Collaborate and Partner Editions of its Orbix E2A Web Services Integration Platform, a comprehensive integration solution that marries Web services standards with powerful business-to-business and enterprise application integration capabilities.



XML Spy 4.3 Developer Tools Released by Altova

(Beverly, MA) – Altova, Inc., has released XML Spy 4.3 Suite, a comprehensive product line of developer tools for

advanced XML application development consisting of the XML Spy 4.3 IDE, XSLT Designer, and Document Editor.

XML Spy 4.3 Suite lets developers tackle real-world Web services development on all of the major Web services platforms including Microsoft .NET and



J2EE. It is best known as a developer tool for testing and debugging Web services based on SOAP.

XML Spy supports graphical editing of XML Schemas that use Microsoft SQLXML schema extensions, a technology that bridges the gap between XML and relational data by creating XML views of relational data, abstracting away the underlying data format.

www.xmlspy.com

XML NEWS

The Collaborate Edition extends the capabilities offered by the Orbix E2A XMLBus Edition with enterprise QoS such as security; packaged application adapters for Baan, PeopleSoft, SAP, and others; and support for emerging Web services standards-based protocols including ebXML and RosettaNet.

The Partner Edition provides small and medium enterprises with a robust, yet easy-to-use platform for business collaboration between trading partners, as well as powerful transformation tools for easy conversion between different data formats.

www.iona.com

Microsoft Introduces BizTalk 2002

(Seattle) – BizTalk Server 2002, with support for orchestrating Web services through Visual Studio .NET, new technology for business partner connectivity, and new tools for management, monitoring, and deployment capabilities within the enterprise, has been announced by Microsoft.

For companies that use complex CRM systems, the BizTalk middleware application offers an extensive packaged adapter library that will allow developers to connect easily with CRM systems as well as ERP, database, and legacy systems.

www.microsoft.com

Software AG Updates Tamino XML Server

(Reston, VA) – Software AG, Inc., has announced the availability of Tamino XML Server, version 3.1. The release includes XML-based access to external database systems, migration tools, Schema editors, a WebDAV server, and interfaces to Java-based applications.

www.softwareagusa.com



BEA Evaluation Version of NeoCore XMS Released

(Colorado Springs, CO) – NeoCore Systems is now offering a BEA evaluation version of its product, NeoCore XML Information Management System (XMS).

The NeoCore XMS Evaluation PAK, BEA Edition plugs directly into and extends the BEA



WebLogic container, providing a tightly integrated and high-performing native XML operational information store. It is available for download to BEA developers at no cost at <http://developer.bea.com:80/tools/utilities.jsp>.

www.neocore.com

Unidex Shipping Latest XML Convert

(Holmdel, NJ) – Unidex, Inc., has released XML Convert 2.2, a Java application that converts legacy data into XML and vice versa.

The new version introduces a SAX2 driver for the XML Convert flat file parser and a SAX2 content handler for the XML Convert flat file builder. XML Convert 2.2 also adds support for nonprintable characters and variable-length nondelimited fields.

XML Convert 2.2 supports a wide variety of flat file formats, including fixed-length fields, delimited fields, CSV, semistructured data (human-readable reports, news feeds, etc.), multiple record types, nested groups of records, and more. It uses XFlat schemas to parse the input data and drive the conversion process. XFlat is a simple XML



language for defining flat file schemas. XML Convert 2.2 can be invoked from the command line or from the user's Java application via SAX or a simple proprietary API.

Visit www.unidex.com/download.htm for an evaluation copy of XML Convert 2.2.

www.unidex.com

ISDA Releases Working Draft of FpML Version 3.0

(New York) – The International Swaps and Derivatives Association (ISDA) has released the Working Draft for Financial products Markup Language (FpML), version 3.0. FpML is the XML-based, freely licensed, e-commerce standard supporting OTC trading of financial derivatives.

Version 3.0 will include coverage of all IRD products previously defined in FpML version 2.0, together with FX



Software AG, Altova Enter Strategic Partnership

(Reston, VA / Beverly, MA) – Software AG and Altova, Inc., have signed a worldwide agreement covering Altova's XML Spy 4.2 Suite and Software AG's Tamino XML Server.

The complementary products give XML Spy users easy and low-cost

access to Tamino, enabling them to natively store and process XML data from new applications. In turn, the partnership extends

Software AG's reach to Altova's community of 400,000 developers.

www.xmlspy.com
www.softwareagusa.com



products such as FX Spot,

Forwards, Nondeliverable forwards, FX Swaps, and FX OTC Options. The full text of the Working Draft is available at www.fpml.org/spec/index.asp. See also www.isda.org for information.

InstantService Updates Customer Contact Tool

(Seattle) – InstantService, a provider of cost-saving chat and e-mail customer service solutions, announced the release of InstantService 3.0, the latest version of its customer contact tool.

The new release includes streamlined Account Administration Utility for faster and easier navigation, re-architecture of



reporting features with new reports, easier navigation and formatting for XML, Language Translation for real-time machine-based translations, and automatic agent logoff after an agent's console becomes inactive for a specified time.

www.instantservice.com

Cross-Media Publishing System Released

(Emigsville, PA) – Vasont, a cross-media publishing system that enables enterprises to publish

content across numerous media channels, has been released by



Progressive Information Technologies, a leader in the information management and publishing services industry for more than 50 years.

Vasont's new features and enhancements include graphical workflow, peer review, dynamic Web applications, content processing, and advanced content modeling. It integrates with industry-leading XML editors and composition tools.

www.vasont.com





XMLisaTreasureMyHeartCannotDeny

It is ironic, and thus divine



Struggling Novelist worked as a Barnes & Noble cashier for six years. There he wrote a fictional memoir, My Life As a Victorian Nobleman, unfortunately still unpublished. He then got a P.R. job at a small book distributor's Web site division. Some suggest that his tech articles are a veiled attempt to be discovered as a novel writer, which he denies.

BY A PRETENTIOUS, STRUGGLING AMATEUR NOVELIST

I tip my hat to you, gentle reader, as I regale you with my technological tales of summers past. I invite you to sit back and enjoy the bounty of my stylesheet memories.

I always prided myself on being a perspicacious fellow. Two seasons ago I decided to learn that which the future would embrace: XML. Ah, yes, I remember it well. I summered in the biggest of apples: New York City! I was an obsequious lad in the service of a petulant Silicon Alley baron. A particularly ruthless pirate, he demanded his corporate soldiers learn all the ways of technological armaments (i.e., he made us learn about the technology our site used). I chose to study the arts of illusion and change, summated in the bastion of ironic logic that is XML.

Indeed, I found many a pleasure in working with XML. My obsession with the immaculate, which some have claimed borders on the pedantic, made XML validating no more a chore than the blossoming of a springtime love! Fie upon those who would say that the rules of XML are too stringent! These men's hearts are as cold as the blackest winter of hate! I shake my fist at the sky over the creation of these enemies of XML! These enemies of love!

I also appreciated the ways in which XML could be a deus ex machina for many a troubled programmer. The catharsis that an XSLT transformation brought about would have brought a tear to the eyes of the most hardhearted stoic.

There are a myriad of magical syntaxes to discover, a virtual ambrosia for the dynamic Web site. The magic of an XML stylesheet

makes data come alive! That which was static and stagnant becomes a delectable, dynamic feast of vibrancy and verve. Forsooth! A fortnight of formatting solutions!

And is it not the ultimate irony that the very machinations of XML are mired in unmovable rules and syntax that do not give way to even a royal man's touch or plea? The strongest and wealthiest suitor would fail to woo the spirit of XML toward a more relaxed set of rules. Yet these very rules become the most vehement agents of limitless possibilities! The banal XML laws are expedient in allowing us untold freedom in creativity...in making page formats.

Some may chide me for my love of XML's ironic nature. To one such as them, I say, taste the back of my hand, rogue! For it is irony that makes the science into the supernatural. The hand that made irony made life itself, for it is everywhere. Is it not ironic that we call people who have a lisp something they can't even pronounce? Is it not irony that we call people who have a stutter something they would struggle with ad infinitum?

I have always said, "All rules have exceptions, without exception, except for this one, which makes it an exception to the rule." Many people recoil when I blurt that irony. How can your eyes not go wide from the flood of theological implications? XML is just as ironic, and thus just as divine. XML has no exceptions. Yet it makes all things possible. Kind of. If there were a modern-day Prometheus, he would at last be at peace. He would know the powers of the mind we have had to unleash to create XML, and he would be proud.

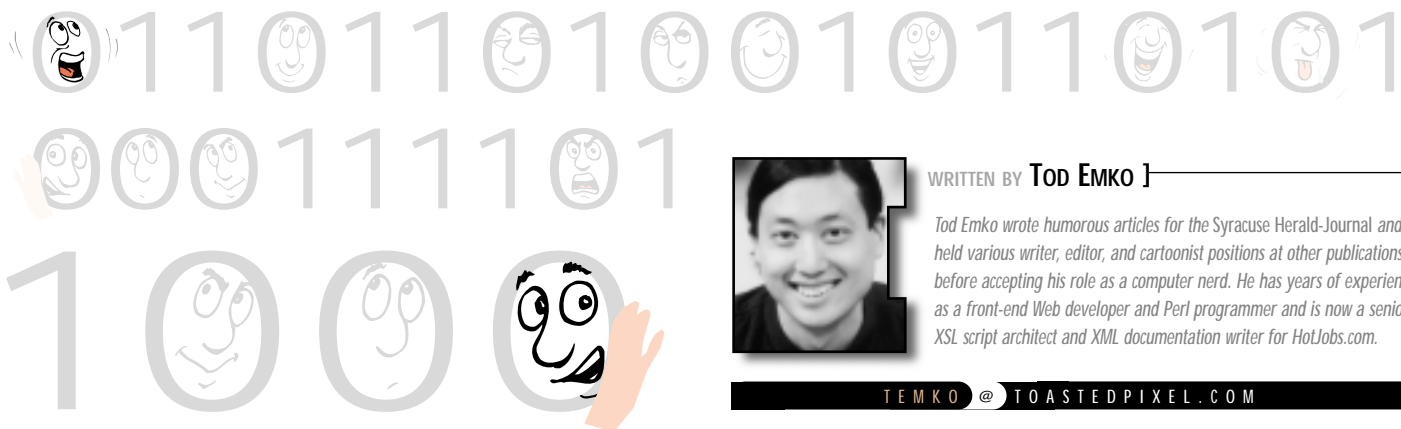
I have slaved to bring together all the wisdom that would bring you technical power. And, in my travels, I found that XML was always the harbinger of power. So hark well to my message, dear reader. XML will bring glory to the humblest house. But use it wisely, and go softly. Do not be corrupted by the total power that comes from the passion you can unleash. But whether you go mad with XML power or rule with XML kindness, I bow to you; it is your world now. I shall rest at last. ☸



WRITTEN BY TOD EMKO }

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

TEMKO @ TOASTEDPIXEL.COM



XML Global

www.xmlglobaltechnologies.com/yourinformation

Altova

www.altova.com