

NEW YEAR BRINGS NEW WAYS TO USE XML

XML JOURNAL

The World's Leading XML Resource

Volume: 3 Issue: 1

XML-JOURNAL.COM



FROM THE EDITOR

Beyond Categories

by Ajit Sagar pg. 5

INDUSTRY COMMENTARY

Soft Issues Surrounding Industry Standard Schemas

by Sean McGrath pg. 7

READER FEEDBACK

Letters to the Editor

pg. 8

PRODUCT REVIEW

Data Junction Integration Suite

by Data Junction

Reviewed by H. Russel Douglas pg. 60

XML NEWS

pg. 64

AND NOW...

I Am an Undefeatable Hipster...

by Tod Emko pg. 66

RETAILERS PLEASE DISPLAY
UNTIL MARCH 31, 2002



SYS-CON
MEDIA

The 1 XML Registry

How to perform B2B transactions!

WRITTEN BY KRISTIAN CIBULSKIS 10

XSLT Tutorial: Got XSLT? Part 3 of 5

Discover how to transform XML to VoiceXML

Shouki Souri 16

Q&A: Shedding a Little Light on XML

Can your XML processor distinguish elements from attributes?

David Silverlight 20

XML Feature: XSL Formatting Objects:

Here Today, Huge Tomorrow Is XSL-FO the 'next big thing'?

Frank Neugebauer 22

XML Tutorial: An Introduction to XSLT Part 1

XML blesses you with a way to separate content from presentation: stylesheets

Steve Hoenisch 30

XML Feature: JAX Pack!

Bridging the gap between Java and XML Technologies

Hitesh Seth 36

XML Feature: SOAP Messages with Attachments

How this emerging W3C note can be used with the Apache SOAP implementation

Ian Moraes 46

XML Security: Using the IBM XML Security Suite

Part 2 Encrypting XML documents dynamically: using the toolkit to perform digital signatures

Joseph Smith 52

XML&Performance: XML Enabled Applications: Need for Speed Here's help when your applications run into bottlenecks

Srinivas Pandrangi 56

Iona

www.iona.com/ws-webcasts

Sonic Software

www.sonicsoftware.com

XML Global Technologies

www.xmlglobal.com/newangle

EDITORIAL ADVISORY BOARD

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

DEPARTMENT EDITORS

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

WRITERS IN THIS ISSUE

Kristian Cibulskis, H. Russel Douglas, Tod Emko,
 Steve Hoenisch, Ian Moraes, Frank Neugebauer,
 Srinivas Pandrangi, Hitesh Seth, David Silverlight,
 Joseph Smith, Shouki Souri

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

For subscriptions and requests for bulk orders,
 please send your letters to Subscription Department
Subscription Hotline 800 513-7111

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

EDITORIAL OFFICES

SYS-CON MEDIA

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

©COPYRIGHT

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

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



**SYS-CON
 MEDIA**



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

from
 the
 editor
 on
 the
 editor

Beyond Categories

Recently I did an analysis of Verticalnet's ontology-based tools. It's interesting that as soon as you say the word *ontology*, people start wondering what they are and what they're good for. I went through a learning process this myself recently and would like to share my findings with you. Ontologies, of course, are closely related to the world of XML.

XML allows authors to create their own markup tags, which seem to carry some semantics. However, from a computational perspective a tag like <BOOK> carries as much in the way of semantics as a tag like <H1>. A computer simply doesn't know what an author is and how the concept "author" is related to, for example, the concept "person."

The challenge in modeling data that implies a richer abstract notion requires a modeling paradigm that is beyond the hierarchical nature of an object tree. This is because a tree limits the relationship between data elements to a parent-child or "is-a" relationship. Basically, a modeling environment that's hierarchical in nature can be used only to define categories of data.

Business applications need very rich data definitions as well as the definition of sophisticated relationships between the data elements being modeled. These relationships need to go beyond the basic parent-child relationship that's used to express categorization. For example, buyers require data to flow from distributors, distributors require data to flow from suppliers, and so on.

Currently, the paradigm to model data exchange of this kind is very limited. It can be achieved using tags that bound the data definition or product-specific tables. XML allows the definition of data tags that can be extended in a hierarchy. However, as the relationships between the data tags become more specialized, XML tends to be too low-level to manage the modeling in an efficient way.

Approaches like ebXML or RosettaNet try to tackle this problem by offering open XML-based frameworks for capturing and expressing electronic business information. However, to map this generic framework to different business environments, you have to map between the terminologies of disparate business environments

The basic idea of an ontology is to define (1) sets of terms relating to a particular business domain and (2) relationships between those terms. The set of terms with the definition of a relationship network constitutes an ontology. For example, a DTD or an XML Schema are examples of primitive ontologies. XML is a great base for expressing ontologies. Languages for expressing ontologies have been built on XML. Examples of such languages are Ontology Exchange Language (XOL), Ontology Markup Language (OML), and Ontology Interchange Language (OIL).

The concept of ontologies is still in its infancy in the realm of actual business applications even though ample research has been conducted on the subject. A good source for more information on ontologies is www.ontology.org. However, to apply this knowledge, standard tools are needed in the market to create ontologies in particular business domains. Verticalnet (www.verticalnet.com) is an example of a company that has a set of ontology-based tools that are currently used for modeling business applications in e-commerce. Another interesting company that I was introduced to at the XML/Web Services Edge Conference & Expo was Unicorn (www.unicorn.com). Unicorn offers tools for mapping XML Schemas.

Although ontologies hold a lot of promise for sophisticated data modeling, several challenges have to be met if this way of modeling is to become a business standard. Industry standards, tools, and education need to become readily available to apply ontologies to problems that exist in e-commerce today.



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 Innovatem, based in Dallas, he is an expert in Java, Web, and XML technologies.

DataMirror

www.datamirror.com/resourcecenter

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

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

ADVERTISING
Senior VP, Sales & Marketing
Carmen Gonzalez carmen@sys-con.com

VP, Sales & Marketing
Miles Silverman miles@sys-con.com
Advertising Director

Robyn Forma robyn@sys-con.com
Advertising Account Manager
Megan Ring megan@sys-con.com

Associate Sales Managers
Carrie Gebert carrie@sys-con.com
Kristin Kuhnle kristin@sys-con.com
Alisa Catalano alisa@sys-con.com

EDITORIAL

Executive Editor
M'lou Pinkham mpinkham@sys-con.com
Managing Editor

Cheryl Van Sise cheryl@sys-con.com
Editor

Nancy Valentine nancy@sys-con.com
Associate Editors

Jamie Matusow jamie@sys-con.com
Gail Schultz gail@sys-con.com

Assistant Editor
Lin Goetz lin@sys-con.com

PRODUCTION

Vice President, Production & Design
Jim Morgan jim@sys-con.com

Art Director
Alex Botero alex@sys-con.com

Associate Art Directors
Cathryn Burak cathyb@sys-con.com

Louis F. Cuffari louis@sys-con.com
Assistant Art Directors

Richard Silverberg richards@sys-con.com
Aarathi Venkataraman aarathi@sys-con.com

SYS-CON EVENTS

VP, Events
Cathy Walters cathyw@sys-con.com

Conference Manager
Michael Lynch mike@sys-con.com

Sales Executives, Exhibits
Michael Pesick michael@sys-con.com

Richard Anderson richard@sys-con.com
Show Assistant

Niki Panagopoulos niki@sys-con.com
Registration Assistant

Jaclyn Redmond jaclyn@sys-con.com

CUSTOMER RELATIONS/JDI STORE
Manager, Customer Relations/ JDI Store

Anthony D. Spitzer tony@sys-con.com
Customer Service Liaison

Patti Del Vecchio patti@sys-con.com

WEB SERVICES

Webmaster
Robert Diamond robert@sys-con.com

Web Designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com

ACCOUNTING

Chief Financial Officer
Bruce Kanner bruce@sys-con.com

Assistant Controller
Judith Calnan judith@sys-con.com

Accounts Receivable
Jan Braidech jan@sys-con.com

Accounts Payable
Joan LaRose joan@sys-con.com

Accounting Clerk
Betty White betty@sys-con.com



WRITTEN BY **SEAN MCGRATH**

Soft Issues Surrounding Industry Standard Schemas

It sounds so easy. First, get a bunch of people together who share a common need to interchange some type of data – say, invoices. Explain the significant technical benefits of having an industry standard schema for invoices.

Get technically minded individuals into a room with plenty of whiteboards and caffeine. Sometime later they'll emerge with a consensus model of what it is to be an "invoice" enshrined in some schema language (UML/XML Schema/DTD/RelaxNG/whatever).

Thereafter, all interested parties use the schema for data interchange and all is sweetness and light.

This makes 100% technical sense but it often doesn't work in the real world. The reasons it doesn't work have nothing to do with flavors of schema language or indeed flavors of markup language. It often doesn't work because of soft issues concerning people.

Let's start with Zipf's law and the Principle of Least Effort (<http://cogsci.umn.edu/millennium/1109153206.html>). Simply put, humans strive to do as little as possible to communicate. The language this article is written in, English, is literally riddled with structures that break the rules of English grammar – the schema – in the interests of quick and easy communication.

All human languages exhibit this phenomenon. Successful languages adapt to allow humans to cut corners in the interest of quicker communications. XML tag languages that don't offer similar functionality are asking for trouble. That trouble can manifest itself in a number of possible ways.

First, users may engage in "tag abuse" – using tags for purposes they weren't intended for because it makes their life easier.

Second, users may create point-to-point side agreements between themselves for simpler communications and convert their simple communications to the official schema only when forced to.

Third, users may lobby for "flexibility" that allows them to make local modifications to the industry standard schema, thus creating an entire family of mutually incompatible but similar languages that start as patois and grow into full-blown, mutually incompatible dialects.

Fourth, users may bring the standard initiative down. If this happens, everything from the coffee to the schema language can be blamed. Everything, that is, except noncompliance with Zipf's Principle of Least Effort.

Then there's the matter of organic schema growth and what I call the "tag bag syndrome."

Successful languages, and XML-based tag languages are no exception, need to exhibit the ability to change and evolve over time. Otherwise they atrophy and eventually die. With human languages we just change their grammatical structures and idioms without worrying about historical material fitting the new forms. In other words, we aren't worried about backward compatibility.

With XML tag languages we're typically very worried about backward compatibility. When we need to modify a schema to cater to a new phenomenon, we cannot allow previously valid documents to become invalid. As a result, we loosen the constraints in the schema. Over time, the gradual loosening of constraints erodes the tight control over structure the schema designers put there in the first place. The result is a bag of tags – structures of the form "X can consist of any number of A or B or C or..." In DTD terminology these are known as *repeating OR groups*.

This leads us nicely to the next soft issue: George Miller's law concerning the cognitive limits of human beings. If you present humans with a list of more than seven options to choose from at any one time, they'll start to feel overloaded and uncomfortable. When it comes to creating XML conforming to a "bag of tags schema," there can be an overwhelming feeling of drowning in a sea of options.

Continued on page 8

SEAN.MCGRATH@PROPYLON.COM

AUTHOR BIO

Sean McGrath is founder and CTO of Propylon, one of Ireland's fastest-growing software companies. Headquartered in Dublin, with development centers in Sligo, Ireland, and Mumbai, India, Propylon delivers what it terms industrial-strength XML and XML consultancy services to its service and product partners in Europe and the U.S.

Letters to the Editor



Reader Questions

I have a couple of questions about Jonathan Knudsen's article in *XML-Journal*, Vol. 2, issue 11 – one about style and one about the specific methodology.

1. In the `commandAction()` function, you choose to program:

```
if (c.getCommandType() == Command.EXIT)
```

where I frequently see

```
if (c == exitCommand)
```

with the addition of code to create the Command "button" in either the constructor or `startApp()`.

I have seen this done both ways, and think that yours would be the most efficient yet less "object" than the other method. Please comment.

2. Again in the `commandAction()` function, you have the previous noted comparison, with the terminate logic, and then the rest of the function is essentially a default case. I understand that this is a very simple example, but these seem to me to be the very bad habits that people get into that eventually turn around and bite them or someone else when it comes time to extend or modify these same functions.

The simple inclusion of a wrapper around the rest of the function:

```
> else if (c.getCommandType() ==
Command.SCREEN) {
>     InputStream rawIn =
>     this.getClass().getResourceAsStream
>     ("/example1.xml");
>     ...
> }
```

makes it very clear to the next person to look at this code that all of this function is not to be performed each time the function is called.

Kevin Timm
via e-mail

Writer Response

Thanks for your thoughtful questions.

1. Using `Command.EXIT` is, perhaps, less object-y. Nevertheless, the appeal is that I'm able to save on a member variable. By testing the command type, I don't have to save a reference to the command itself, which simplifies my class a little bit.
2. Fair enough – this is mostly a question of style. I live in my own little world, but if I were writing code that I thought someone else would have to maintain someday, I might well make it more explicit as you suggested.

Jonathan Knudsen
jonathan.knudsen@sun.com

Simplicity Is the Key

I read with delight Mark Baker's article on

"Designing XML Tagging Languages" (*XML-J*, Vol. 2, issue 8). I don't use OmniMark, but I do try to create document structures and XSL that are as simple as will do the job. In fact, I try not to use a formal DTD until I have done some early functional prototypes for a customer.

Thanks for the article.

Dorothy J. Hoskins
via e-mail

Where, O Where?

What happened to Ketan Patel's series? I thought it was going to be a 4-part series. Part 1 ("Business – Not Dialects – Will Drive XML Adoption") was in September, and Part 2 ("XML, Collaboration, and Workflow") was in October, but I can't find Part 3 in the November issue. Will there be another article?

I really like the series, and I'm looking forward to reading more.

George Cummings
via e-mail

I'm glad you like the series. It will resume in our February issue. 🌀

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

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

Industry Commentary *Continued from page 7*

In the back office the software developer is similarly overwhelmed because the large number of choices at each point in the schema translates into a programmer's worst nightmare, known as an *exploding state space*.

Schema creators ignore Miller's rule at their peril. User and software developers alike can and will subvert the technically elegant designs in order to work within comfortable cognitive limits.

It's important to remember that the "group of people getting together to agree on a data interchange standard" phenomenon is not a new one. Stepping back just one generation, this went on wholesale in the SGML world – without much success. It wouldn't be good soft-issue tactics for me to mention any failed standards initiatives by name. You'll find plenty of evidence if you read through some history of the SGML years, especially the early-to-mid-'90s.

My contention is that the failed industry-standard schema ini-

tiatives of the past did not fail for technical reasons; they failed for human reasons. There is a rich lore of experience here that the new wave of XML schema designers could do worse than mine for valuable insights. Those that don't learn from the mistakes of the past truly are doomed to repeat them.

Apart from paying due regard to history, I think XML schema design needs to take a leaf out of the extreme programming book. Start with the customer (human), do the smallest thing that can possibly work, and start using it. Never lose sight of the human creating the XML content or the human writing software to process the content. Remember Miller's law. Remember Zipf's law. Allow the schema to grow/evolve organically over time. It will anyway, whether the schema designer likes it or not.

The only industry standard schemas still standing at the end of this decade will be the ones that address the soft issues. 🌀

HitSoftware

www.hitsw.com



The 1 eb

Want to perform B2B transactions? This registry will help

THE ELECTRONIC BUSINESS EXTENSIBLE MARKUP LANGUAGE, BETTER KNOWN AS EBXML, AIMS TO ALLOW COMPANIES OF ANY SIZE TO CONDUCT BUSINESS ELECTRONICALLY VIA THE INTERNET. OBVIOUSLY, COMPANIES DOING BUSINESS TOGETHER ISN'T A NEW IDEA. EDI (ELECTRONIC DATA INTERCHANGE) HAS BEEN USED BETWEEN LARGE BUSINESSES TO CONDUCT ELECTRONIC BUSINESS SINCE THE 1960s. HOWEVER, EDI OFTEN REQUIRES THE IMPLEMENTATION OF CUSTOM PROTOCOLS AND PROPRIETARY MESSAGE FORMATS BETWEEN THE INDIVIDUAL COMPANIES.

WRITTEN BY
KRISTIAN CIBULSKIS

XML Registry

Because of this, its use has been restricted to larger corporations that can absorb the initial costs required to do business in this fashion. The goal of ebXML is to provide a flexible, open infrastructure that will let companies of any size, anywhere in the world, do business together.

The ebXML effort is jointly sponsored by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and OASIS, the Organization for the Advancement of Structured Information Standards, along with approximately 30 other industry leaders. UN/CEFACT is also the standards body behind EDIFACT, an EDI standard used heavily throughout Europe and the Pacific Rim.

The ebXML group has delivered three key components of a next-generation B2B infrastructure:

- An XML messaging specification
- A trading partners agreement specification
- A registry/repository specification

A second initiative at OASIS has begun to create a Universal Business Language (UBL), essentially a standard set of XML business documents to be used for B2B transactions. UBL is based on xCBL 3.0, which is freely available and widely deployed.

In this article we'll explore the ebXML Registry/Repository, one of the cornerstone components of the ebXML architecture.

What Is the ebXML Registry?

The ebXML Registry serves as a central repository that enables businesses to share information. The ebXML Registry Services specification defines it as "a set of services that enable sharing of information between interested parties for the purpose of enabling business process integration between such parties based on the ebXML specifications." So, in addition to being a directory of content, it's a storage mechanism. Essentially, it's a place where people can locate, store, and retrieve objects with the intention of performing B2B transactions. Figure 1 illustrates the role of the registry in a typical B2B scenario.

What types of objects does the registry handle? Actually, the Registry Information Model (RIM) is described in its entirety in another specification (see **Resources** section). At its core are two key objects: the *RegistryEntry* and the *ClassificationNode*.

The *RegistryEntry* object is meant to contain an ebXML-specified object. For example, a Collaboration Protocol Profile (CPP), a Collaboration Protocol Agreement (CPA), a UBL document, or even a software component could be held within a *RegistryEntry*.

ClassificationNodes are used to create tree structures that are then used to define Classification schemes or ontologies. Most of the other registry objects are associated with one or more *RegistryEntries* and are described in Table 1.

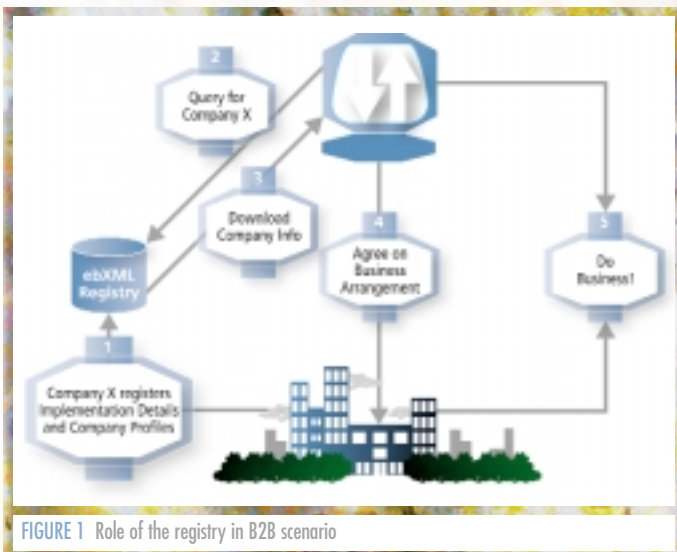


FIGURE 1 Role of the registry in B2B scenario

Classifications

Before we discuss how to interact with an ebXML Registry, it's probably worthwhile to discuss the concept of Classifications. In an ebXML Registry a classification hierarchy exists independent of any objects to be classified. A classification hierarchy is described by a collection of ClassificationNodes.

For example, we may have a "Geography" classification tree, as depicted in Figure 2. This classification tree simply describes the hierarchy of possible classifications. For example, it indicates that "Korea" is a member of "Asia." Both Korea and Asia are ClassificationNodes.

When an object is stored in the registry, it can be associated with a number of ClassificationNodes. For example, a company submitting a document for a company in Korea could associate that document with the Korea ClassificationNode. This is called a *Classification*. Then, when someone queries for all the companies in Asia, the Classification hierarchy is used to determine that any entry with an association, or Classification, directly to the "Korea" ClassificationNode is also indirectly Classified by "Asia."

When operating with an ebXML Registry, it's important to remember the distinction between a ClassificationNode and an actual Classification. Both can be created, queried, and modified, so you need to know which one you actually intend to work upon.

The Public Interfaces

How do we access the objects described above? The ebXML Registry specification calls for two basic interfaces for interaction with the registry: the ObjectManager interface and the ObjectQueryManager interface. A UML diagram of these interfaces can be seen in Figure 3. The ObjectManager interface provides life-cycle methods for the objects maintained by the registry while the ObjectQueryManager interface provides methods for locating and retrieving objects from the registry.

While the specification doesn't explicitly describe an implementation technology, it does describe the exact functions provided by each of these interfaces. However, the recommended implementation technology is actually SOAP and XML. So every time you read something about a given method with certain parameters, think of a SOAP request as the method with an XML payload that represents the parameters.

The ObjectManager Interface

An object in an ebXML Registry can exist in one of four basic states. Figure 4 depicts the various object states in the registry and the ObjectManager methods that cause the various state transitions.

When an object is first introduced to the registry it exists in a Submitted state. From there the object can be moved into an Approved state. Once an object has reached that state, it's ready for use by the participating business parties. Objects in the Approved state can continue to be updated and modified. A time may come, however, when this object

needs to be retired. At that time the object can be moved to a Deprecated state. Once in this state, the object continues to be accessible but may no longer be modified or updated. Eventually it may be moved to a Removed state where it is no longer accessible by the participating business parties. Effectively, the object has been removed from the registry.

The ObjectManager interface provides the methods to create new objects and affect state transitions on existing objects. However, all of these methods are not available to the general public. We'll take a look at the security restrictions placed on these methods later.

The ObjectQueryManager Interface

Now that we know how to create objects within the registry, we need a way to find and access those objects. That is the purpose of the methods provided via the ObjectQueryManager interface. Just as SELECT statements constitute the majority of requests processed by a database, these query methods are the most common requests of an ebXML Registry. In that light it accounts for why over 50% of the content of the ebXML Registry specification has been devoted to the query interfaces!

To support the various types of queries that users may want to execute, three different query methods have been defined. Any registry that claims to be ebXML compliant must support the first two, known as the Browse and Drill Down Query and the Filter Query. The third is a SQL Query interface, which isn't required by the ebXML specification. However, the specification makes it clear that if you do support a SQL Query type interface, it must meet the specifications described in the ebXML Registry specification. It's an optional, yet standardized, interface. Let's take a look at each of these in detail:

Browse and Drill Down Query

As you might imagine, the simplest type of query interface is an interactive query in which the user provides the "filtering" capabilities. The Browse and Drill Down Query of an ebXML Registry provides this capability. Three methods provide this functionality.

The first one is `getRootClassificationNodes`, which basically returns all ClassificationNodes in the registry that don't have a parent. The only parameter that can be specified, `namePattern`, works like a SQL-92 LIKE clause, allowing you to specify a wildcard pattern for the query.

Now that you have a collection of root ClassificationNodes, you may want to "drill down" into one of these nodes by retrieving all of its children. This is accomplished through the `getClassificationTree` method, which has two parameters, `parent` and `depth`. `Parent` is the parent for which the children are being retrieved. `Depth` is an optional element. If it isn't specified, it defaults to 1, and only the immediate children of the given parent are retrieved. Otherwise, `depth` refers to how many levels of the hierarchy of a given parent should be retrieved. A value of zero or less indicates that all children should be retrieved.

NAME	DESCRIPTION
Package	Packages are used to group together related RegistryEntries
ExternalLink	Allows for external items not managed by the registry to be associated with a RegistryEntry; for example, a link to the submitting company's home page could be included
Classification	Associates this particular RegistryEntry with a ClassificationNode
Slot	Generic data containers that can be dynamically added and removed from a RegistryEntry; this provides the ability to add arbitrary attributes to a RegistryEntry
ExternalIdentifier	Used to hold additional identifier information for given RegistryEntry, such as a DUNS number, social security number, or tax ID number
AuditableEvent	Collection of AuditableEvents that describe the audit trail for a given RegistryEntry

TABLE 1 ebXML Registry objects

Macromedia

www.macromedia.com/downloads

Once users have browsed their way down to a *ClassificationNode* that they're interested in, they'll most likely want to look at the *RegistryEntries* associated with that *ClassificationNode*. This request is made through the *getClassifiedObjects* method which is supplied with a list of *ClassificationNodes* that the user is interested in. The registry then returns all objects that are either directly classified by the supplied *ClassificationNode* or directly classified by a descendant of the supplied *ClassificationNode*, and thus indirectly classified by the parent *ClassificationNode*. If multiple *ClassificationNodes* are supplied, they are AND'ed together. That is, the object returned must be classified, either directly or indirectly, by each of the supplied *ClassificationNodes*.

It's easy to see how the Browse and Drill Down Query interface could be used to build an interactive GUI. This is exactly why it was created: to facilitate the creation of ebXML Registry browsers that a business could use to locate a particular service.

Filter Query

While the Browse and Drill Down Query interface is well suited for building a GUI, it isn't very powerful. To support more complex queries, the Filter Query interface is provided, which supports an XML syntax that describes a set of class filters. Each of these class filters is a predicate clause intended to restrict the result set.

The topmost restriction when performing a Filter Query pertains to the type of object you're looking for. For example, you may be interested in finding a *RegistryEntry*, an *AuditableEvent*, or a *ClassificationNode*. From there you specify various class filters to restrict the result set. A working example may be the best way to describe how this Filter Query interface operates. Take a look at the sample in Listing 1, which is taken from the ebXML Registry Specification.

- **Line 1:** This identifies this request as a request for a *RegistryEntry*. Other possible types of queries include the *AuditableEventQuery*, the

call to the *submitAdhocQuery* method of the *ObjectQueryManager* interface.

As you can see, the Filter Query provides the capability to execute rich and complex queries against an ebXML Registry. This query interface is mandatory for any ebXML registry. However, many people are familiar with a different query language, namely SQL. To provide a query interface that's already known by the masses, the ebXML specification also specifies an optional SQL Query interface.

SQL Query

A given ebXML Registry implementation may choose to support the optional SQL Query interface. This interface supports a basic subset of the SELECT statement as described by the SQL-92 standard. It's also been extended to allow for the calling of stored procedures. Although the ebXML Registry specification does supply a sample relational database schema for data storage, the data can be stored in any manner and still be ebXML compliant.

The specification essentially defines a binding between the RIM and a set of fictional database tables that are used in a given SQL Query statement. For example, if you wanted to find the global ID of any *RegistryEntry* with a name containing "Java" and a version number other than 2, you could issue the following SQL Statement to the SQL Interface:

```
SELECT id
FROM RegistryEntry
WHERE name LIKE '%Java%'
AND majorVersion > 2
```

When querying *ClassificationNodes*, the primary identifier is a unique ID created by the registry. Using these IDs you can query other nodes by parents and so forth. However, an alternative identifier has also been pro-

The Browse and Drill Down Query interface facilitates creation of ebXML Registry browsers that a business could use to locate a particular service

ClassificationNodeQuery, the *RegistryPackageQuery*, the *OrganizationQuery*.

- **Lines 2-4:** This describes the first filter that's placed on our return items. In this case we place a restriction on the status attribute of the *RegistryEntry* in which the status must be equal to "Approved." A full set of string operators, such as EQUAL, STARTSWITH, and ENDSWITH, as well as numerous other operators for boolean and numeric values, exist.
- **Line 5:** This section begins a filter on the Classifications associated with this *RegistryEntry*.
- **Lines 6-9:** In this first section we specify that there must be a *ClassificationNode* that starts with, or has a parent identifier of, urn:ebxml:cs:industry. This indicates that we are placing a restriction on the industry of the result set. The next predicate furthers that restriction to the "Automotive" industry through an XPATH-like identifier.
- **Lines 10-13:** Similar to the previous clause, this clause restricts us to a particular geography, namely Japan.

Once the XML describing your query has been created, it can be executed by making a

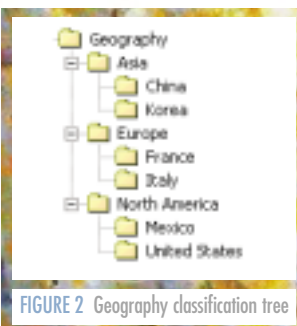


FIGURE 2 Geography classification tree



FIGURE 3 ObjectManager and ObjectQueryManager interfaces

vided for use in SQL Queries. Each *ClassificationNode* can also be located by an XPATH-like expression describing this node. An example path for a *ClassificationNode* would be something like "/Geography/Asia/Japan/Tokyo".

To submit a SQL Query, the text of the SQL statement is passed as a parameter to the *submitAdhocQuery* method of the *ObjectQueryManager*, just as a Filter Query would be submitted.

Content Retrieval

The query methods we've examined don't return the actual content we're interested in. Rather, they return a unique identifier of the *RepositoryEntries* that we want to retrieve. The only thing left to do is to retrieve our actual data. To do this, we simply call the *getContent* method and supply a list of identifiers for the objects we wish to retrieve. If there are no errors in retrieving the data, a success response is returned along with the requested data. Each requested object is returned as a separate, additional payload. Keep in mind that you can store any type of data in the registry – not only XML, but other document types, object models, and even software components.

Security

While the Registry Information Model is capable of supporting quite a sophisticated security model, the current ebXML Registry specification calls for a minimalist approach to security. The basic philosophy is that "Any known entity can publish content and anyone can view published content."

To meet the above requirement, the specification defines three different types of users: RegistryGuest, ContentOwner, and RegistryAdministrator.

The RegistryGuest role describes any unauthenticated use of the registry. According to the basic security philosophy, anyone can view published content. Therefore there's no need to authenticate a large portion of the user community. A RegistryGuest should be able to call any read-only, getXXX(), methods on any registry objects.

The other two roles, however, require authentication to the registry. Since the registry doesn't support the concept of a session containing multiple messages, each message is treated as an independent request that potentially needs to be attributable to a particular principal. The specification requires that a registry use "a credential-based authentication mechanism based on digital certificates and signatures." Given an authentication mechanism, we can now determine who is making a particular request, if necessary, and which role that user has in the registry.

The ContentOwner role is implicitly granted to someone who is the submitter or owner of registry content. ContentOwners can access all of the methods available on the objects that they own. This includes the life-cycle methods of the ObjectManager interface. The RegistryAdministrator role is a super-

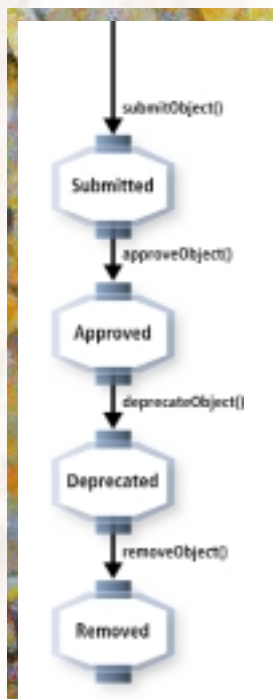


FIGURE 4 Object states and ObjectManager methods

deploying on the iPlanet Application Server, it can be deployed on any J2EE application server.

The second path is to access an external ebXML Registry hosted by another company. Two companies currently working on public ebXML servers are the Korea Trade Network and the DISA Registry Initiative (DRIVE). The Korea Trade Network provides an openly accessible registry service. However, most of the site is in Korean, so if you can't read it I suggest using something like AltaVista's Babel Fish to translate the pages in real time. The DISA Registry is currently under implementation. However, as ebXML gains momentum, more public ebXML Registries will begin to appear.

The Future

Several companies are working to provide commercial implementations of the ebXML Registry and other core technology components of ebXML. Over the next six to 18 months the UBL group will be producing business schemas that will be freely available to the public. While the ebXML and UBL technologies are still in their early stages, they have a huge amount of support in the industry. Given this amount of support and the clear opportunities in B2B transactions between companies of any size, this technology is one to watch this year.

Resources

- **ebXML home page:** www.ebxml.org
- **OASIS home page:** www.oasis-open.org
- **Registry specification:** www.ebxml.org/specs/ebRS.pdf
- **Registry information model:** www.ebxml.org/specs/ebRIM.pdf
- **Using UDDI to find ebXML registries/repositories:** www.ebxml.org/specs/rrUDDI.pdf

While the ebXML and UBL technologies are still in their early stages, they have a huge amount of support in the industry

user role assigned to the administrator of the registry. A RegistryAdministrator is able to call any method on any Registry object.

While the current registry specification describes a minimalist security model, the actual RIM can support a much richer security model. In the future, the Registry specification will most likely support more complex ACL and role-based security for individual objects.

What About UDDI?

While some people believe that the ebXML Registry and UDDI are in competition, they are actually complementary. A key distinction between the two that must be understood is that while UDDI contains a directory of references to Web services information, the ebXML Registry contains the objects themselves. Actually, the two technologies are best used in conjunction. A company interested in finding a new ebXML business partner would first use UDDI to locate a particular ebXML Registry/Repository. This search would return a reference to an ebXML Registry. The company could then access the ebXML Registry to discover the CPP for their new potential partner and then initiate a new CPA.

Implementations

If you're interested in gaining some hands-on experience with an ebXML Registry, there are two basic paths you can take. The first is to download and install an implementation of an ebXML Registry. Sun has a free J2EE implementation of the ebXML Registry specification available for download (see Resources section). Although it includes instructions for

- **Sun's ebXML registry/repository implementation:** www.sun.com/software/xml/developers/regist
- **Korea Trade Network:** www.ebxml.or.kr/registry/index.html
- **AltaVista Babel Fish:** <http://world.altavista.com>
- **DISA registry initiative:** www.disa.org/drive

AUTHOR BIO

Kristian Cibulskis is a freelance consultant focusing on Enterprise Java and XML solutions.

K C I B U L @ H O T M A I L . C O M

LISTING 1

```
1: <RegistryEntryQuery>
2:   <RegistryEntryFilter>
3:     status EQUAL "Approved"
4:   </RegistryEntryFilter>
5:   <HasClassificationBranch>
6:     <ClassificationNodeFilter>
7:       id STARTSWITH "urn:ebxml:cs:industry" AND
8:       path EQUAL "Industry/Automotive"
9:     </ClassificationNodeFilter>
10:    <ClassificationNodeFilter>
11:      id STARTSWITH "urn:ebxml:cs:geography" AND
12:      path EQUAL "Geography/Asia/Japan"
13:    </ClassificationNodeFilter>
14:  </HasClassificationBranch>
15: </RegistryEntryQuery>
```

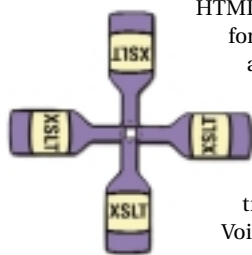
DOWNLOAD THE CODE @
www.sys-con.com/xml



Transform an XML example to speech

GotXSLT?

Part 3 of 5



Someday soon you'll be able to tell the Internet what you're interested in and have it respond with the information you need. You could instruct a Web site to monitor traffic reports, for example, then tell it where you're heading. If there's a traffic jam, the Web site will dial your cell phone to alert you and suggest an alternate route.

VoiceXML is the technology that will make this happen.

In Parts 1 and 2 of this series (*XML-J*, Vol. 2, issues 10, 11) I briefly discussed how to transform an XML document into HTML (Web) using client-side transformation with Internet Explorer 5.0 and using XSLT to transform an XML document into WML (WAP).

In this tutorial I'll discuss how, using the same framework we developed in Part 2, XSLT can transform our XML example to VoiceXML (speech). The primary purpose of the tutorial, however, is to introduce the following:

- The basics of VoiceXML 2.0
- The use of XSLT to transform XML to VoiceXML

Since this is only a very basic introduction to VoiceXML, we won't be focusing on developing VoiceXML applications, but rather the transformation. In fact, the resultant VoiceXML is a very simple document that doesn't prompt the user for any inputs. If you want to learn more about VoiceXML 2.0 and how to develop advanced features, please check the **Resources** section at the end of this tutorial.

What Is VoiceXML?

VoiceXML is an XML-based markup language that defines a spoken dialog just as HTML defines a graphical Web page. It was developed by the VoiceXML Forum as a consolidation of several proprietary formats developed earlier by the forum founders – AT&T, IBM, Lucent, and Motorola. The VoiceXML 1.0 specification was published in March 2000. It did not, however, define speech grammar or audio format, leaving this to

each vendor to implement, thus limiting the ease of portability of VoiceXML applications. Actually, I developed a few 100% VoiceXML 1.0-compliant applications using one of the voice vendors but they wouldn't work on another voice vendor. Talk about defeating the purpose of having specifications!

Fortunately, on October 23, 2001, the World Wide Web Consortium published the VoiceXML 2.0 standard. This new standard is expected to be adopted quickly in the voice industry, and new voice-based applications will be designed predominantly in accordance with VoiceXML. Major actors in the voice industry are backing the VoiceXML 2.0 standard specified by the W3C subgroup for VoiceXML.

In fact, the VoiceXML Forum has signed a Memorandum of Understanding with the W3C marking the Forum's release of the trademark to the public domain. The signing of the MOU coincided with the W3C release of the first Public Working Draft of VoiceXML version 2.0.

The 2.0 specification is based on extensive industry experience with VoiceXML 1.0. The principal differences:

- VoiceXML 2.0 requires support for the W3C speech grammar, audio, and speech synthesis languages. It includes a new log element and makes obsolescent the dtmf, emp, div, pros, and sayas elements. There are many changes to other elements and clarifications that will ensure greater interoperability.
- The specification has been clarified in many areas and reorganized for ease of understanding.

An application written in VoiceXML defines the prompts to be uttered by the system, the speech recognition grammars, and the actions the system is to

take based on what the user says. In many ways these capabilities are similar to those of a speech-enabled application in a traditional Interactive Voice Response environment. In fact, to an end user, spoken access to the Internet using a VoiceXML application will appear very similar to accessing an IVR-based speech-enabled application.

How Does VoiceXML Technology Differ from IVR Systems?

VoiceXML has at least three advantages over proprietary voice platforms:

1. **It's open platform:** VoiceXML was developed with an open platform in mind. There's no need to use proprietary tools or run on proprietary platforms. In contrast, some speech applications (including IVR) were developed using proprietary tools and run on proprietary platforms. Changing platforms is extremely labor-intensive and expensive. VoiceXML provides an open environment with standardized dialog scripting and speech grammar formats that developers have to learn only once.
2. **You can leverage existing Web integration:** VoiceXML allows back-end development to be done once for any Web environment, spoken or graphical. In addition, back-end development is done with standard Web development techniques for both client- and server-side development, such as CGI scripts, Java servlets, and JSP. This allows application developers to leverage existing Web development skills rather than having to learn proprietary systems.
3. **It's declarative and simple:** VoiceXML attempts to make the description of the dialog declarative. This means that the developer describes the dialog that is to be implemented at a relatively high level. The voice browser platform han-

AUTHOR BIO

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

dles the low-level details of transitioning between dialog states, activating speech recognizers, handling telephony, and playing prompts. Most current application development environments are procedural in that they require the developer to handle these details manually, through programming in languages like C++, Java, Visual Basic, and proprietary IVR scripting languages.

What Do We Need?

This tutorial requires the use of the following software and services:

- **Xalan-java version:** www.apache.org
- **JDK 1.2 or 1.3:** www.javasoft.com (I'm assuming you've already installed JVM on your machine)

You also need to register with Tellme Studio at <http://studio.tellme.com> (the service is free).

Our Original XML Document

Listing 1 shows our original XML document (mybooks.xml). We won't reference any XSL stylesheet onto our XML document because we'll plug in our VoiceXML, XSL, and XML files to perform the translation from a command line.

The Goal

Our goal is to produce a VoiceXML document and upload it as in Figure 1.

Creating the Stylesheet Document

Let's devise a stylesheet to convert the mybooks.xml document to VoiceXML. Listing 2 shows a complete XSL file (mybooks2voice.xsl) to transform our XML file (mybooks.xml) into a VoiceXML file (mybooks_vxml.vxml). Listing 3 shows a completely transformed VoiceXML file.

Performing the Transformation from the Command Line

As explained in Part 2, you must include xalan.jar and xerces.jar on your system class path to perform the translation. Java.org.apache.xalan.xslt.Process provides a basic utility for performing transformations from the command line. The command line for most standard transformations is:

```
java org.apache.xalan.xslt.Process -
in xmlSource -xsl stylesheet -out
outputfile
```

where "xmlSource" is the XML source file name, "stylesheet" is the XSL stylesheet file name, and "outputfile" is the output file name. If you want the output to be displayed on the screen, simply omit the -out flag and argument.

I've saved both my mybooks.xml and mybooks2voice.xsl in the d:\gotxslt directory. To perform the translation, type:

```
D:\>cd gotxslt
D:\gotxslt>java
org.apache.xalan.xslt.Process -in
mybooks.xml -xsl mybooks2voice.xsl
-out mybooks_vxml.vxml
```

If all goes well, a new file, mybooks_vxml.vxml, will be the output from the XML to VoiceXML translation. This file is a pure ASCII file, so it can be opened and studied in any text editor.

Using/Configuring Tellme Studio

Once you register with Tellme Studio, click on "MyStudio." On this page are two tabs, Application URL and Scratchpad. Click on Application URL tab. Enter the URL for your VoiceXML file (mybooks_vxml.vxml). For example:

```
http://www.mywebserver.com/voicexml/
mybooks_vxml.vxml
```

Obviously, you need to upload this file to your Web server for the Tellme application to see your file. You can modify the Application URL for your application at any time from your MyStudio page by updating the information and clicking "update".

Note: If you're going to use Geocities (Yahoo's free service) as your Web server, you need to change the extension from ".vxml" to ".xml" in order to upload the file. This should be okay since VoiceXML is an XML file.

Don't Have a Web Server?

Want to test some simple VoiceXML without having to save it to a Web server? Just click and copy the VoiceXML document, then click Scratchpad tab and paste it in. You can also click on the "check syntax" button to see if any errors occurred during the copy/paste procedure.

Now all you have to do is call 1-800-555-VXML to preview it – provided that you entered your developer ID and password. That's all there is to it.

Understanding Our VoiceXML Output

Now let's take a closer look at our mybooks_vxml.vxml. We won't go over the stylesheet because it's simple and so easy to understand. Instead, we'll go over the output VoiceXML file (see Listing 2) to familiarize ourselves with VoiceXML syntax.

Since VoiceXML belongs to the XML family, the first line of the XML declaration is obligatory. After some PI (pro-



FIGURE 1 VoiceXML Tellme Studio

cessing instruction) for the stylesheet, we match the root element, <BOOKS>, which defines the top-level (root) element for the document.

Next we have to let the voice browser know that we're conforming to VoiceXML 2.0. Currently, VoiceXML provides two types of dialogs: <form> and <menu>. The first provides information for the user or presents fields to be filled by the user input. Every <form> dialog contains one or more form items, which are elements within a form that describe some kind of user interaction related to filling in the form.

The <menu> element presents a list of choices to the user and transitions to the chosen information. The <menu> is a convenient syntactic shorthand for a form containing a single anonymous field that prompts the user to make a choice and transitions to different places based on that choice.

Moving further into the document, the first form item we see is <block>. A <block> is a kind of executable context that provides a set of commands to be executed sequentially. Blocks are usually used for presenting information to the user. Within the <block>, the <prompt> and <audio> are used.

Note: The Tellme Voice Application Network integrates a third-party text-to-speech (TTS) engine to convert text to speech. Formerly, Tellme used Lernout & Hauspie RealSpeak for TTS support. Tellme now uses AT&T Natural Voices to perform this task. Per Tellme and AT&T, "The AT&T Labs Natural Voices TTS engine increases the natural quality of synthetic speech so that customers can comfortably listen for longer periods of time – without fatigue or annoyance." In other words, Tellme switched to AT&T Natural Voices for its superior speech synthesis capabilities.

The switch may introduce some incompatibilities that require you to

modify the TTS strings in your application. Learn about the specific differences at <http://studio.tellme.com/general/tts.html>.

In our example we used the TTS engine for all the prompt elements, but you can also use an audio element inside the prompt element to specify a file containing recorded audio. The <audio> elements are also using the TTS engine. The <audio> element plays a prerecorded audio file or text that's converted to synthesized speech. If the src or data attribute points to a valid audio file, then any text specified is ignored. If the audio file isn't found, the specified text is played to the user. If you specify a URL or location of your .wav files to a nonexistent file and don't provide alternate text, you get a TTS error message.

A prerecorded voice offers better audio quality than a synthesized voice. Therefore it is always recommended that you add audio files even if you don't have any.

We use the <break> element, which stops the speech for the amount indicated in milliseconds.

Notice that we did not define any grammar in this tutorial. The reason is that our example is so simple that it didn't prompt the user for any input. A grammar defines the set of valid expressions that a user can say when interacting with a voice application. Each interactive dialog in an application references one or more grammars using one or more grammar elements.

Conclusion


This article gives you an easy way to generate a VoiceXML file manually by using an XSLT processor. There are, of course, several other ways that you can generate VoiceXML or any files dynamically using XSLT and Java servlets as an example.

I hope that you have gained valuable knowledge concerning the use of XML/-XSLT technologies. I also hope it's

enough to get you excited about XML technology and get you thinking about where it's going. Both XML and XSLT are currently receiving a lot of interest from vendors and developers alike due to their simplicity and flexibility in customizing the content. As for VoiceXML, I am a firm believer in this technology.

Resources

- **VoiceXML 2.0 W3C:** www.w3.org/TR/2001/WD-voicexml20-20011023/
- **VoiceXML Forum:** www.voicexml.org/mou.html
- **Tellme VoiceXML 2.0 features:** <http://studio.tellme.com/features/feature2001-10-23.html>
- **Apache XML Project:** <http://xml.apache.org/>

You can also register with the Tellme studio, which is freely available at <http://studio.tellme.com>. 

SSOURI @ PANAMSAT.COM

LISTING 1 XML document from Part 1 tutorial

```
<?xml version="1.0" ?>
<BOOKS>
  <BOOK>
    <ISBN>1-565-92869-9</ISBN>
    <CATEGORY>Programming</CATEGORY>
    <RELEASE_DATE>2000-03-21</RELEASE_DATE>
    <TITLE>Enterprise JAVABEANS</TITLE>
    <PRICE currency="US">34.95</PRICE>
  </BOOK>
  <BOOK>
    <ISBN>1-861-00311-0</ISBN>
    <CATEGORY>XML</CATEGORY>
    <RELEASE_DATE>2000-03-10</RELEASE_DATE>
    <TITLE>PROFESSIONAL XML</TITLE>
    <PRICE currency="US">49.99</PRICE>
  </BOOK>
  <BOOK>
    <ISBN>0-596-00016-2</ISBN>
    <CATEGORY>XML</CATEGORY>
    <RELEASE_DATE>2000-07-11</RELEASE_DATE>
    <TITLE>JAVA and XML</TITLE>
    <PRICE currency="US">39.95</PRICE>
  </BOOK>
</BOOKS>
```

LISTING 2 "mybooks2voice.xsl" — Complete XSL file to transform XML into VoiceXML

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/">
    <xsl:apply-templates select="BOOKS"/>
  </xsl:template>
  <xsl:template match="BOOKS">
    <vxml version="2.0">
      <form id="book_form">
        <block>
          <prompt>
            <audio>My Book Catalog</audio>
            <break time="1000ms"/>
            <xsl:for-each select="BOOK">
              <audio>Book number <xsl:number/></audio>
              <audio>Title is <xsl:value-of select="TITLE"/></audio>
              <audio>ISBN number is <xsl:value-of select="ISBN"/></audio>
              <audio>Price is $<xsl:value-of select="PRICE"/></audio>
            </xsl:for-each>
          </prompt>
        </block>
      </form>
    </vxml>
```

```
<block>
  <prompt>
    <audio>Done with book list</audio>
    <break time="300ms"/>
    <audio>Good Bye</audio>
  </prompt>
</block>
</form>
</vxml>
</xsl:template>
</xsl:stylesheet>
```

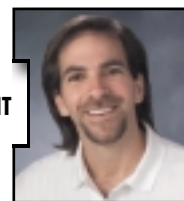
LISTING 3 "mybooks_vxml.vxml" — Complete VoiceXML file

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
  <form id="book_form">
    <block>
      <prompt>
        <audio>My Book Catalog</audio>
        <break time="1000ms"/>
        <audio>Book number 1</audio>
        <audio>Title is Enterprise JAVABEANS</audio>
        <audio>ISBN number is 1-565-92869-9</audio>
        <audio>Price is $34.95</audio>
        <break time="500ms"/>
        <audio>Book number 2</audio>
        <audio>Title is PROFESSIONAL XML</audio>
        <audio>ISBN number is 1-861-00311-0</audio>
        <audio>Price is $49.99</audio>
        <break time="500ms"/>
        <audio>Book number 3</audio>
        <audio>Title is JAVA and XML</audio>
        <audio>ISBN number is 0-596-00016-2</audio>
        <audio>Price is $39.95</audio>
        <break time="500ms"/>
        <audio>Done with book list</audio>
        <break time="300ms"/>
        <audio>Good Bye</audio>
      </prompt>
    </block>
  </form>
</vxml>
```

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

BEA eWorld

www.bea.com/events/ewrld/2002



Is your XML processor psychic?

Shedding a Little Light on XML

I got an e-mail the other day from Miss Cleo. It was quite exciting. She said she had a dream about me and that I needed to call her (or one of her psychic friends) to find out some important information about my future. Wow! I thought. So I ran to the phone.

We chatted for some time and I found out my girlfriend was cheating on me. How did she know? Those psychics amaze me. While I was moving out I thought how much this relates to XML namespaces.

XML, as you know, allows us to work with XML documents that have many different vocabularies. Elements and attributes may have identical names from different document models in a single XML document. Unless your XML processor has psychic abilities, it will need some additional information to distinguish between these elements and attributes and to understand their meaning correctly. This is where namespaces come into play.

Although the namespace recommendation has made this task easier for us, there are still a few areas of confusion. This month's column covers – and sheds a little light on – some of the very common questions on namespaces.

Q: *Why do namespaces use a URL naming convention if they don't reference a Web site?*

A: The Web has enjoyed the success of utilizing a naming convention that allows for unique file names regardless of the enormous number of files on the Internet. It's amazing if you think about it, yet still no problem with name resolution. If this concept had been considered before the Internet blossomed, the question of how to make certain that file names are unique would have been a huge challenge. Today it's simply a matter of creating a domain for your "corner of the Web." Most of us take this for granted.

A URL-formatted string is the most common form of a URI used for namespaces. URI strings are

used in the W3C Namespace recommendation ("Namespaces in XML 1.0" at www.w3.org/TR/1999/REC-xml-names-19990114) to guarantee uniqueness in element type and attribute names when mixing vocabularies.

Any URI can be used for a namespace URI string, and the URL form is one of those allowed strings. The protocol identifier at the start of the URI dictates how the rest of the URI string is interpreted. The protocol identifier "http:" indicates that the next portion of the string is a domain name, and the "ownership" of domain names is governed. The remainder of the URI string can be any valid sequence of characters chosen by the owner of the domain. Provided that no one uses someone else's domain without permission, uniqueness is guaranteed between users.

What if a single document contains two elements with the same name? It's possible to use two different vocabularies within the same document, with element-type names used for different

purposes. How can we distinguish them? This is precisely the problem that namespaces were created to solve.

This is also where the URL naming convention comes into play. The philosophy behind this is that URL naming convention has allowed us to define unique Internet filenames; why not use them to enforce unique element names in our XML documents? Similar problem? Yes. Similar solution? Yes. Similar success? Perhaps.

Well, this demonstrates a double-edged sword. Essentially, we've been conditioned to expect a resource at the end of the URL and first-time users find it a source of great confusion when looking for that "XML Library" or some other resource at the location specified by the URL. Take, for example, a namespace defined in an XSLT document:

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

On the surface it may appear that this namespace reference is pointing to the foregoing URL for reference to a possible library, a class, or even a definition of the current XSL spec. This is a common cause of confusion because we're conditioned to expect to retrieve *something* when we see a URL.

Admittedly, I made this invalid assumption myself at first. Yes, it's true, and I'm not afraid to admit it! Following is an excerpt of a conversation I had when I first started out:

Me: Are you *sure* it doesn't point to any type of library or anything required to make my XML work?

Expert: Yes.

Me: You're not lying to me, are you?

Expert: No.

Me: Why would it use a URL if it doesn't point to anything?

Expert: URLs are great at ensuring globally unique names.



AUTHOR BIO

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



One of the best things about XML is the ease with which data can be shared between applications



Me: Will it fail if I'm not connected to the Internet at the time? [I thought I had him on this one.]

Expert: No, your document doesn't actually look at the resource. It just needs to follow that naming convention.

Me: Can I have a cup of coffee now?

Expert: Yes, I think you'll need it.

Bottom line: In Namespaces 1.0 the specification of URI syntax allows for globally unique names.

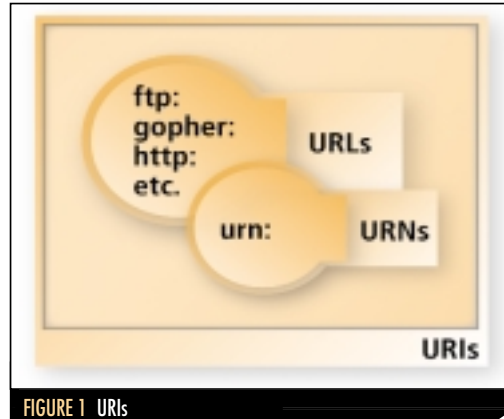


FIGURE 1 URIs

Q: What's the difference between a URL, a URN, and a URI?

A: Well, we're all too familiar with URLs, but a few other flavors of the UR* family have cropped up recently. Although they share a related goal of identifying resources, each has some definite distinctions. Following are some TLAs (three-letter acronyms). Figure 1 demonstrates how they're related.

- **URI:** Uniform Resource Identifier
- **URL:** Uniform Resource Locator
- **URN:** Uniform Resource Name

A **URI** is the broadest form of the term to describe a resource. In practice, the terms **URI** and **URL** are interchangeable; the primary distinction is that a **URI** merely identifies a resource whereas a **URL** identifies it and describes where and how to retrieve it. A **URI**, in fact, is a more generalized version of a **URL**, since it can refer to both a specific resource and a relative resource. You'll also find that certain characters permitted in **URIs** aren't permitted in **URLs**.

From Figure 1 you can see that **URLs** are a subset of **URIs**. Referring to a resource as a **URL** is a bit of an antiquated reference, yet it still gets the point across for users who are new to the UR* family. Actually, you might be revealing that you're technically over the hill if you refer to a resource as a **URL** in a technical document. **URLs** are most commonly used in conjunction with a subset of technologies such as **http**, **ftp**, and **gopher**. This first part of a **URL**, referred to as the *scheme*, tells an application what protocol to use in order to retrieve the resource.

A **URN** is a globally unique, location-independent resource name. The location-independent nature is especially important here. If you change the name of the server where your **URN** resources are located, you won't break all of the links associated with it the way you would with a **URI** or **URL**. Much the same way that an **ISBN** number is associated with identifying a book, if you change the location of a book the **ISBN** number won't change and you can still find it.

Microsoft's Hailstorm and .NET suite of products make great use of **URNs** and I suspect we'll see much more of them in the future.

Q: Why should I use namespaces in my XML?

A: That's a good question and it has many answers. The "X" in XML stands for *extensible*. Without namespaces, extending an XML vocabulary is a difficult task. It's easy to add your own custom documentation tags to XSLT because your namespace won't collide with the XSLT namespace and confuse whatever XSLT engine you're using.

Another good reason is that one of the best things about XML is the ease with which data can be shared between applications. Without namespaces, sharing data becomes next to impossible. Applications have to make assumptions about the nature of the data they receive rather than rely on the namespace to identify data they can

handle and data they can't or don't need to process.

(This could be a whole article in and of itself.)

Q: Where are namespaces headed?

A: Although the URI string in Namespaces 1.0 is *not* used for resource discovery, considerations are underway to publish a future version of Namespaces in which the URI string can be used for such a purpose. The Resource Directory Description Language (RDDL - www.rddl.org/) provides a text description of classes of resources, and of individual resources within those classes. RDDL is suitable for describing schemas, style-sheets, executable code, or any other resource associated with a vocabulary associated with a namespace URI. It has been proposed to embed RDDL descriptions in an XHTML document dereferenced from the namespace URI providing both human-readable and machine-readable information about a class of documents that are instances of namespace-identified vocabularies.

Elements of Design

While we're on the topic of namespaces, I recently discovered a great utility for maintaining, editing, and tracking XML namespaces. This tool, Namespaces Navigator, gives you the ability to add, edit, and track all the namespaces in your development environment.

The editing environment is very accommodating as it allows you to view and/or update any section of your namespace. Pretty cool stuff. I'm glad somebody thought of this. What's more, it's built on a framework that allows for building entire apps on XML, XSLT, and HTML. The framework is another intriguing aspect of this tool and can be an educational process all on its own. Namespaces Navigator is actually an application of this framework. You can check it out at www.topxml.com/itse.

DSILVERLIGHT @ INFOTERIA.COM

XSL FORMATTING OBJECTS: HERE TODAY,

WRITTEN BY FRANK NEUGEBAUER

As those of you familiar with XSL know, there are two parts to the W3C Recommendation (www.w3.org/TR/xsl/): a transformation part (XSLT) and a formatting part (XSL Formatting Objects, or XSL-FO for short) with the intent being the presentation of XML. However, since XSLT is also its own (more mature) W3C Recommendation (www.w3.org/TR/xslt), it has enjoyed the attention of developers wishing to transform XML into other markup languages such as HTML. In a very real sense XSLT is how XML is currently being visually presented.

Although using XSLT to transform XML to HTML can be very powerful and useful, it also has some serious limitations. For example, HTML, even when combined with Cascading Style Sheets (CSS), can cause unpredictable results when printing. Another limitation is the need for developers to understand the inner workings of the expected eventual output format(s) (e.g., HTML, WML) and to code XSL Stylesheets for each such output format expected.

In theory (and partially in practice), XSL Formatting Objects can overcome these shortcomings because the language is a kind of “general” markup language with extensive formatting capabilities without being output format-specific. It may eventually be positioned as the ultimate language for expressing the output format of XML documents across software (e.g., Web browsers, word processors) and hardware (e.g., printers, cell phones) boundaries.

Admittedly this speculation is a bit optimistic – the result of the promise brought about by the growing maturity and capability of XSL-FO processors. Currently, a number of maturing XSL-FO processors are capable of rendering such formats as Portable Document Format (PDF), Rich Text Format (RTF), and plaintext, among others. An extensive list of available XSL-FO processors can be found at www.w3.org/Style/XSL/ (check out the far left side of the page under “Implementations”).

In this article I'll use a demo version of one such processor, RenderX's (www.renderx.com) XEP, to demonstrate how XSL-FOs work in the hopes of providing you with the same “enlightenment” I experienced when I first started using XSL-FO.

It's important to note that, with perhaps the exception of XSL-FO tools writers, XSL-FO shouldn't be used alone (like HTML) but rather in conjunction with XSLT to express the output format of XML. However, for clarity, I'm going to assume that you have some familiarity with XSLT and thus will leave out those details for the most part. It's also important to note that XSL-FO consists of many Formatting Objects (tags) and properties (attributes on those tags); thus this article is but a short primer on the topic. I encourage you to search the Web for more information about XSL-FO.

Before I continue, I'd like to offer some formal definitions of acronyms, since XSL, XSLT, and XSL-FO are sometimes referred to interchangeably even though they're not necessarily interchangeable:

- *XSLT* exists as its own W3C Recommendation and can be used alone when referring to the process of transforming XML into something else (e.g., XML, HTML, plaintext).
- *XSL-FO* exists only as part of XSL and shouldn't be used alone (recall that, although you can express XSL-FO by itself, FO documents should really be created through XSLT).
- *XSL* therefore implies that both XSLT and XSL-FO are involved. Figure 1 shows the conceptual model of XSL.

HUGE TOMORROW

IS XSL-FO THE 'NEXT BIG THING' TO PRESENT XML DATA?

What is XSL-FO, anyway? Formally speaking, it's the intended language for expressing the output format of XML documents and is part of the recently adopted W3C XSL 1.0 Recommendation. XSL-FO (actually, XSL as a whole) was intentionally influenced by the Document Style Semantics Specification Language (DSSSL) and the CSS language. In fact, wherever possible, the names and meanings of CSS attributes are kept intact within XSL-FO.

The XSL-FO language itself is actually expressed as well-formed XML that uses the "fo" namespace ("www.w3.org/1999/XSL/Format"). An XSL-FO document can exist as a file (saved with a .fo or .xml extension) or as a stream, and is plaintext. As alluded to previously, the tags of the

XSL-FO markup constitute the Formatting Objects, and the (mostly) CSS-like properties constitute attributes on those tags. Listing 1 depicts a simple XSL-FO document.

This basic document was created by hand, but should normally be the output from XSLT. The basic hierarchical structure of the FO in this document is depicted in Figure 2. Listing 1 (and the figure) represents a minimum FO document structure. I should mention that for the sake of brevity several other optional elements are missing.

I'll provide more detail about how these elements are laid out in FO documents and their relationships to one another in the sections below.

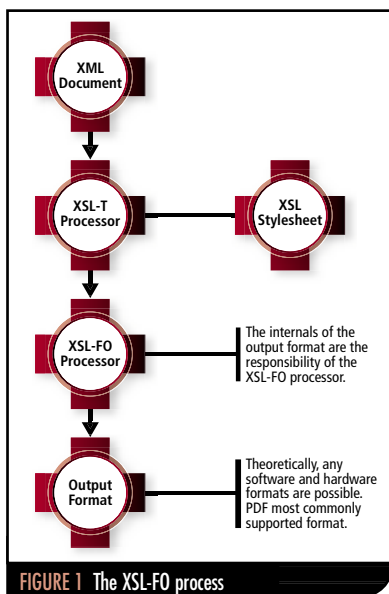


FIGURE 1 The XSL-FO process

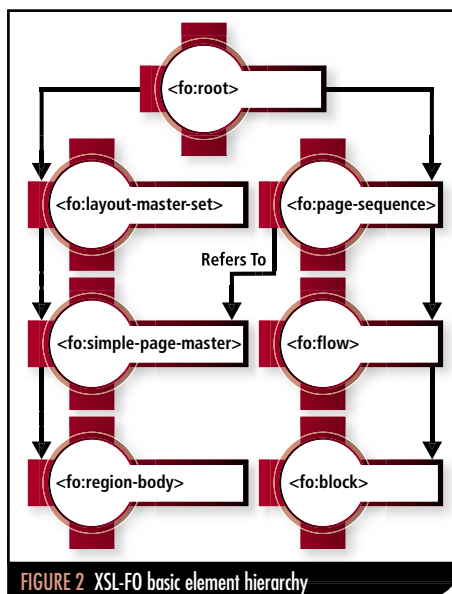


FIGURE 2 XSL-FO basic element hierarchy

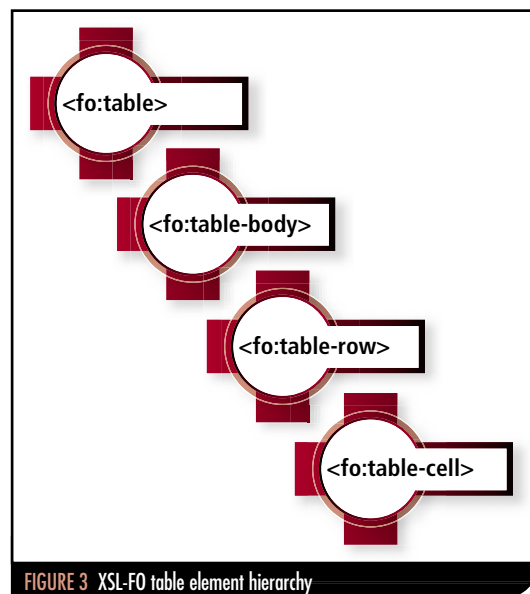


FIGURE 3 XSL-FO table element hierarchy

TABLES IN XSL-FO ARE DEFINED USING NESTED ELEMENTS

Page Setup

<fo:root>

As a well-formed XML document, FO must have a single root element. In FO this element is `<fo:root>`, and although it contains a “media-usage” property per the Recommendation, I don’t believe this property is used in current implementations. Note that in Listing 1 the FO namespace is contained as an attribute. If you were to create an FO document by hand, this declaration would be necessary. However, because the document should be the result of XSLT, the namespace declaration would actually be part of the `<xsl:stylesheet>` declaration in the XSL stylesheet.

<fo:layout-master-set>

The `<fo:layout-master-set>` element can be thought of as the container for page definitions. These definitions take the form of `<fo:simple-page-master>` and/or `<fo:page-sequence-master>` elements. I’m not going to cover the latter element except to say that it can dictate how `<fo:simple-page-master>` elements are generated. Refer to the XSL Recommendation for more information about `<xsl:page-sequence-master>`. The fact that you can define multiple pages within a single FO document is an important distinction of XSL-FO when compared to HTML. No attributes are associated with this element.

<fo:simple-page-master>

The `<fo:simple-page-master>` element is the basic definition of a page. Later within an FO document, `<fo:page-sequence>` elements can refer to a particular page-master (through the “master-name” attribute contained within each) to specify how “this” page should look. The `<fo:simple-page-master>` element can take responsibility (through attributes) for the following parts of a page definition:

Page Margins

These are controlled through the `margin-left`, `margin-right`, `margin-top`, and `margin-bottom` attributes. A “shortcut” for margins (“margin”) will set the four margins to the same specified value. For example:

```
<fo:simple-page-master margin="1in">
```

The value of this attribute is generally set in inches (“in”) or centimeters (“cm”).

Page Orientation/Size

This is controlled via the “page-height” and “page-width” attributes. Also generally set are either inches or centimeters, setting them appropriately will yield the orientation (i.e., landscape or portrait) and size of the page. For example, for a standard portrait page in inches:

```
<fo:simple-page-master page-height="11in" page-width="8.5in">
```

Page Divisions

XSL-FO provides for five different regions of a page: “region-body,” “region-before,” “region-after,” “region-start,” and “region-end.” Each of these regions serves important geometric roles within a page. The “region-before” represents the top section of a page and is used for things like headers. Similarly, “region-after” represents the bottom section of a page and is used for footers (e.g., page numbering). The “region-start” and “region-end” represent the left- and right-hand sides of a page, respectively. For now, the “region-body” is of particular importance and serves to specify the main section of a page.

There are two parts to specifying the “region-body.” The first part is to specify the `<fo:region-body>` element within an `<fo:simple-page-master>` element as in Listing 1. The second part is to refer to the “xsl-region-body” value of the “flow-name” attribute of the `<fo:flow>` element (again, see Listing 1). This establishes that a particular “flow” is to be rendered within the “body” of the page.

<fo:page-sequence>

The `<fo:page-sequence>` element is the specification for a particular type of page definition within an FO document. I use the term *page definition* because there isn’t necessarily a relationship between the number of `<fo:page-sequence>` elements and the number of pages in the output. Rather, an `<fo:page-sequence>` specifies that all content within this `<fo:page-sequence>` will look a particular way as defined by its attribution and (optionally) static content (I’ll demonstrate static content shortly; its most common use is for things like headers and footers).

The `<fo:page-sequence>` element has several attributes, but the only one required is the “master-name” element, which refers to one of the `<fo:simple-page-master>` elements defined within the `<fo:layout-master-set>`. This sets the “terms” (e.g., orientation, margins) that the content within the `<fo:page-sequence>` will abide by.

<fo:flow>

"Body" (i.e., nonheader/footer) content within an FO document is contained within <fo:block> (I'll examine this element shortly) elements that must be contained within flowing regions defined by the <fo:flow> element. One important constraint involving the <fo:flow> element: only one <fo:flow> element can be contained within a single <fo:page-sequence> element. You can use the attribute "flow-name" to specify in which of the five regions (specified as "xsl-region-body," "xsl-region-start," "xsl-region-end," "xsl-region-before," and "xsl-region-after") you want to place the flow, but you can have only one <fo:flow> within a given <fo:page-sequence>. So, for example:

```
<fo:flow flow-name="xsl-region-body">
```

This line implies that the flow will appear as part of the "body" of the document.

<fo:block>

The basic building-block content within FO documents is the <fo:block>. This element may appear within several elements, including <fo:flow>, <fo:static-content>, <fo:table-cell> (for tables), <fo:list-item-label> (e.g., the label in lists), and <fo:list-item-body> (the content of lists). Not to mention, static text can also be contained within <fo:block>. Many attributes are defined for <fo:block>, and include "text-align," "text-indent," "font-size," "font-family," "font-weight," and "color." For example:

```
<fo:block font-family="sans-serif" font-size="10">Some text</fo:block>
```

This markup creates a block of text with the 10-point sans-serif font and contains the text "Some text."

<fo:static-content>

Although not part of Listing 1 or Figure 2, I'd like to cover the <fo:static-content> element because of its importance in creating page content. This element is commonly used for repeating content within pages – for example, headers and footers. Its lone attribute, "flow-name," specifies where within the document the content will appear. The values "xsl-region-before" and "xsl-region-after" specify that the content should appear as a header or footer, respectively.

Within this element several elements may be contained. For example, to specify a running page total, an <fo:block> is created within which the <fo:page-number> element is placed as follows:

```
<fo:static-content flow-name="xsl-region-after">
  <fo:block>
    Page Number: <fo:page-number/>
  </fo:block>
</fo:static-content>
```

An important fact about this element is that if one or more <fo:static-content> elements also appear within the same <fo:page-sequence>, they must appear before the <fo:flow> element (recall that there can be only one <fo:flow> per <fo:page-sequence>).

Creating Tables with XSL-FO

Tables within XSL-FO are defined using a series of nested elements and are created somewhat similarly to HTML tables. Figure 3 illustrates the basic hierarchy of FO tables.

Content within tables exists with <fo:block> elements contained within the <fo:table-cell> element. Listing 2 gives a complete example of an FO document containing a simple table. Figure 4 shows this table as a PDF file when processed by the demo version of RenderX's XEP FO processor.

Listing 2 is a minimum set of markup required to create a table in XSL-FO. A number of attributes available on the elements specified in Listing 2 (and Figure 3) allow control over tables. I've left out those details because of space constraints, but I encourage you to check the XSL Recommendation for more information. One thing I didn't leave out was the font embellishment, but note that such attributes actually exist on the <fo:block> element within the <fo:table-cell> element and aren't actually part of the table.

Creating Lists with XSL-FO

Like tables, FO lists are created using a hierarchy of FO elements. Figure 5 shows the hierarchy of XSL-FO elements that make up lists. Lists in XSL-FO consist of two parts: a label, which can be represented as any <fo:block>, including text and/or graphics; and the body, which is also represented as an <fo:block> and can contain any kind of content.

The only tricky part of setting up lists in XSL-FO is setting up the margins and spacing between the label and the body of the list item. Figure 6 shows the various XSL-FO elements and the attributes used to set up lists. I've separated the XSL-FO element from the attribute with a colon (":").

At a minimum, the "provisional-distance-between-starts" and "provisional-label-separation" attributes of the <fo:list-block> element should be defined to specify the distance between the starts of the label and body and the distance between the end of the label and the start of the body. The thing to remember is that the smaller the "provisional-label-separation," the more room that the label can take. So, in general, to set up the list, determine how large your label is and how much room you want between the label and body. Assume, for example, that you have a label that will be 1 in. wide (text as opposed to a bullet or number) and you want .25 in. between the end of the label and the start of the body. At a minimum, your "provisional-distance-between-starts" should

Identification	Item Name	Price (USD)
139439	Monitor	\$524.00

FIGURE 4 XSL-FO table output as PDF

FO LISTS ARE CREATED USING A HIERARCHY OF FO ELEMENTS

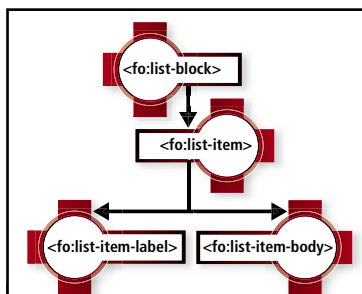


FIGURE 5 List hierarchy in XSL-FO

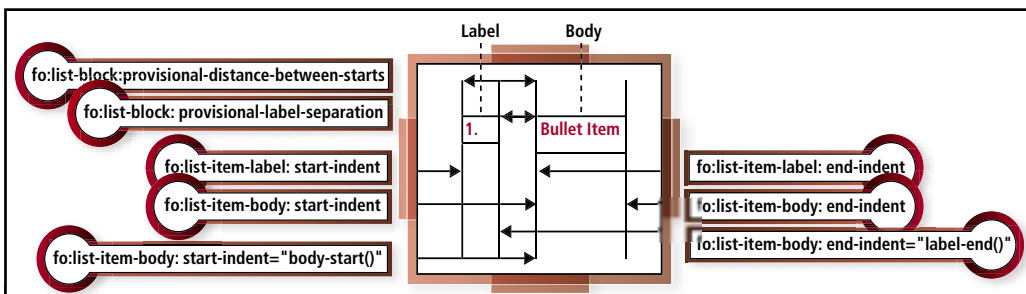


FIGURE 6 XSL-FO list margins

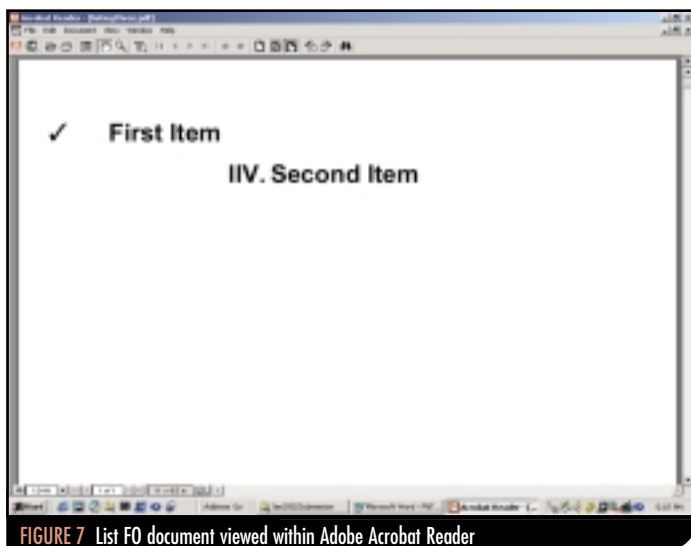


FIGURE 7 List FO document viewed within Adobe Acrobat Reader

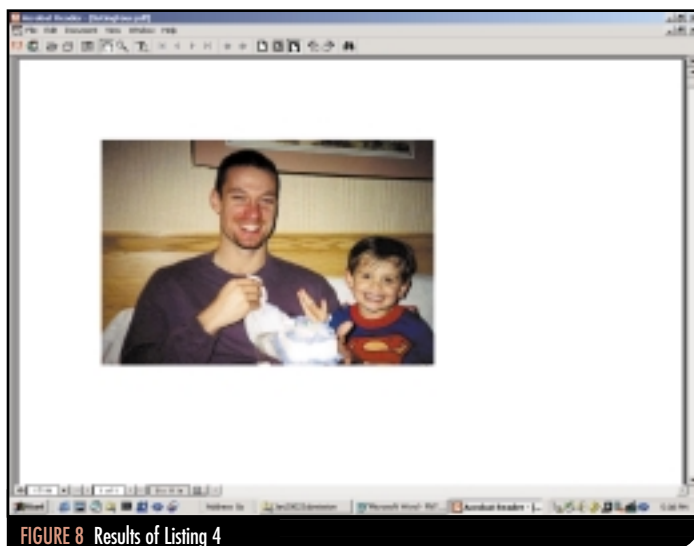


FIGURE 8 Results of Listing 4

XSLT IS IDEALLY SUITED TO TRANSFORM XML INTO XSL-FO

be 1.25 in. (the sum of the label and the separation between the label and body). I should note, however, that this kind of calculation is far from science, since FO processors have internal algorithms to adjust text. Nevertheless, this rule is a good starting point.

The last two items in Figure 6 (`start-indent="body-start()"` and `end-indent="label-end()"`) are XSL-FO "shortcuts" (they're actually functions) used to automatically set the start of the body equal to the end of the label (calculated by using the values from the `provisional-distance-between-starts` and the `provisional-label-separation` attributes contained within the `<fo:list-block>` element). Using these functions is applicable only in situations where a margin isn't required. The reason is that margins are set (e.g., for a left-indented list) by using the `start-indent` attribute of the `<fo:list-item-label>` and `<fo:list-item-body>` elements.

Listing 3 depicts a complete FO document with two lists. The first is a typical list with a bullet (represented using the Zapf Dingbats font) that starts from the left margin. The second is an atypical list with a margin. Notice, in particular, how the second list doesn't use `provisional-distance-between-starts` or `provisional-label-separation`. The reason is that the margins are set manually using the `"start-indent"` attribute of the `<fo:list-item-label>` and `<fo:list-item-body>` elements. Figure 7 shows this document after being processed by XEP.

Embellishments

Throughout this article I've shown sample XSL-FO syntax with a minimum of embellishments. However, it's important to understand that XSL-FO contains a considerable array of techniques for making XML text stand out in a crowd.

Recall that XSL-FO uses a CSS-like syntax. For example, consider the `<fo:block>` within Listing 1. If you modify that block as follows:

```
<fo:block font-color="red">A simple fo document</fo:block>
```

the text, when rendered, will be the color red, as expected. Note that the attribute was set on the `<fo:block>` element, a common, but not the only, place to add embellishment. That's not the only way color can be used. Other examples include setting the background color (by using the `"background-color"` attribute) for tables, blocks, and entire output documents. You can use color similarly when setting border colors.

The font weight (`"font-weight"`), size (`"font-size"`), and family (`"font-family"`) can also be altered to add pizzazz to your documents. (Examples of font embellishments can be seen throughout this article). The best way to learn how to use this aspect of XSL-FO is through experimentation guided by the XSL Recommendation.

Extras

Many documents require images. Consider the mass proliferation of graphics within Web pages, magazine articles, and books. Without functionality similar to this, XSL-FO would lack a key ingredient. Fortunately, the `<fo:external-graphic>` element is available to insert images into documents.

The `<fo:external-graphic>` works a lot like the HTML `` tag. It's generally contained within an `<fo:block>` with attributes to specify the image source (`"src"`), height (`"content-height"`), width (`"content-width"`), and a host of other (perhaps expected) properties of inserted images. How `<fo:external-graphic>` is supported by FO processors varies. Thus it's best to consult your particular processor's documentation. As an example, Listing 4 depicts a simple FO document that imports a graphic image. To get the image imported, I placed it within the same directory as I ran the processor; RenderX's XEP took care of the rest. The results of running this document through the XEP product are shown in Figure 8 (that's my son Frank and I a few years ago).

Using All of XSL

As I've stated several times, XSL-FO was intended to be the result of an XML transformation – XSLT. I'd be remiss if I didn't include a complete example showing how this is accomplished. Recall (refer to Figure 1) that "real" XSL starts with an XML document and an XSL stylesheet that are fed into an XSLT processor (I'm going to use Apache's Xalan for XSLT). The result is an XSL-FO document that's fed into an FO processor (e.g., RenderX XEP, Apache FOP) that outputs the specified output format (e.g., PDF). The XML document in this example is the code snippet below.

```
<?xml version="1.0"?>
<SimpleRoot>
  <Content>A simple FO document</Content>
</SimpleRoot>
```

Listing 5 shows the XSL stylesheet used to transform the XML into Listing 1. That is, this example is simply going to externalize the text "A simple FO document" into an XML document that is then transformed into an FO document just like the one in Listing 1.

In Listing 5 the namespaces for both XSL and FO are declared within the `<xsl:stylesheet>` element. Everything else is basic XSLT; a template matches on the root element of the XML (`<SimpleRoot>`) and then an `<xsl:value-of>` element is used to place the value of the `<Content>` element into the result tree. The remainder of the stylesheet is simply creating the required FO elements. This is similar to the way most transformations from XML to HTML take place using XSLT. The result of this trans-

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-BizXML>, XML & Business, XML Demystified, XML & E-Commerce, XML Middleware, and much more!

XML-J The Complete Works
Reg \$59.99

Buy Online
Only **\$53.99**

COLDFUSION Developer's Journal

The most complete library of exclusive CFDJ articles!

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

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

CFDJ The Complete Works
Reg \$79.99

Buy Online
Only **\$71.99**

CF Advisor

The most complete library of exclusive CFA articles!

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

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

CFA The Complete Works
Reg \$79.99

Buy Online
Only **\$71.99**

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



JDJStore.com

Order Online and Save 10% or More!

WWW.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

formation can then be fed into an XSL-FO processor to create a PDF.

The point I'm attempting to illustrate here is simply that instead of "composing" FO documents as I've done throughout this document, XSLT is ideally (and intentionally) suited to transform XML into XSL-FO and is what XSL is all about.

• • •

I'd like to conclude this article with some speculation regarding XSL. I believe XSL has the potential to eliminate the need to learn markup languages such as HTML and at the same time provide a mechanism to display XML in the same way across hardware and software boundaries.

Imagine FO processors built by vendors such as Microsoft that output XSL-FO documents to the Office suite of tools and Internet Explorer. How about another created by Palm that creates a Palm Pilot output format? Or one created by HP that outputs to its printers?

If these types of FO processors are created, we as stylesheet writers can simply express how we want our XML to be presented and leave the actual rendering details to the vendors. This is not unlike the current FO processor implementations that create PDF, RTE, MIF, and printer output formats. For the record, I have no influence or "insider" information regarding whether or not Microsoft, Palm, HP, or anyone else is ever going to create such FO processors. But wouldn't it be great if they did? ☺

AUTHOR BIO

Frank Neugebauer is a consultant with the Insurance Solutions Group of IBM Global Services. He's been using Java since 1996 and has worked on the architecture and implementation of enterprise Java solutions using servlets, EJBs, XML, and XSLT.

NEUGGS @ HOTMAIL.COM

LISTING 1 Simple XSL-FO document

```
<?xml version="1.0"?>
<fo:root
  xmlns:fo="http://www.w3c.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="body">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-name="body">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>
        A simple FO document</fo:block>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
```

LISTING 2 Simple XSL-FO table document

```
<fo:root
  xmlns:fo="http://www.w3c.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master
      master-name="main"
      page-height="11in"
      page-width="8.5in"
      margin="1in">
      <fo:region-body margin-top=".5in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-name="main">
    <fo:flow flow-name="xsl-region-body">
      <fo:table>
        <fo:table-body>
          <fo:table-row>
            <fo:table-cell>
              <fo:block
                font-size="20pt"
                font-family="sans-serif"
                font-weight="bold"
                line-height="22pt">
                Identification
              </fo:block>
            </fo:table-cell>

            <fo:table-cell>
              <fo:block
                font-size="20pt"
                font-family="sans-serif"
                font-weight="bold"
                line-height="22pt">
                Item Name
              </fo:block>
            </fo:table-cell>

            <fo:table-cell>
              <fo:block
                font-size="20pt"
                font-family="sans-serif"
                font-weight="bold"
                line-height="22pt">
                Price (US)
              </fo:block>
            </fo:table-cell>
          </fo:table-row>
        </fo:table-body>
      </fo:table>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

```
</fo:table-row>

<fo:table-row>
  <fo:table-cell>
    <fo:block
      font-size="18pt"
      font-family="sans-serif"
      line-height="20pt"
      space-after.optimum="15pt">
      138438
    </fo:block>
  </fo:table-cell>

  <fo:table-cell>
    <fo:block
      font-size="18pt"
      font-family="sans-serif"
      line-height="20pt"
      space-after.optimum="15pt">
      Monitor
    </fo:block>
  </fo:table-cell>

  <fo:table-cell>
    <fo:block
      font-size="18pt"
      font-family="sans-serif"
      line-height="20pt"
      space-after.optimum="15pt">
      $524.00
    </fo:block>
  </fo:table-cell>
</fo:table-row>
</fo:table-body>
</fo:table>
</fo:flow>
</fo:page-sequence>
</fo:root>
```

LISTING 3 Simple list FO document

```
<?xml version="1.0"?>
<fo:root
  xmlns:fo="http://www.w3c.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master
      master-name="Page"
      margin=".5in"
      page-height="8.5in"
      page-width="11in">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-name="Page">
    <fo:flow flow-name="xsl-region-body">
      <fo:list-block
        font-weight="bold"
        margin-top=".5in"
        provisional-distance-between-starts="1.0in"
        provisional-label-separation=".20in">
        <fo:list-item>
          <fo:list-item-label
            end-indent="label-end()">
          <fo:block
            font-family="ZapfDingbats"
            font-size="30">&#10003;
          </fo:block>
        </fo:list-item>
      </fo:list-block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

```

</fo:block>
</fo:list-item-label>

<fo:list-item-body
  start-indent="body-start()">
  <fo:block font-size="30">
    First Item
  </fo:block>
</fo:list-item-body>
</fo:list-item>
</fo:list-block>

<fo:list-block font-weight="bold">
  <fo:list-item>
    <fo:list-item-label
      start-indent="3.0in">
      <fo:block font-size="30">
        IIV.
      </fo:block>
    </fo:list-item-label>

    <fo:list-item-body
      start-indent="3.7in">
      <fo:block font-size="30">
        Second Item</fo:block>
      </fo:list-item-body>
    </fo:list-item>
  </fo:list-block>
</fo:flow>
</fo:page-sequence>
</fo:root>

```

LISTING 4 Images using XSL-FO

```

<?xml version="1.0"?>
<fo:root
  xmlns:fo="http://www.w3c.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master
      master-name="body"
      margin="1in">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>

```

```

</fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-name="body">
  <fo:flow flow-name="xsl-region-body">
    <fo:block>
      <fo:external-graphic
        content-height="2in"
        content-width="2in"
        src="image.jpg"/>
    </fo:block>
  </fo:flow>
</fo:page-sequence>
</fo:root>

```

LISTING 5 'True' XSL stylesheet

```

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

  <xsl:template match="SimpleRoot">
    <fo:root >
      <fo:layout-master-set>
        <fo:simple-page-master
          master-name="body">
          <fo:region-body/>
        </fo:simple-page-master>
      </fo:layout-master-set>

      <fo:page-sequence master-name="body">
        <fo:flow flow-name="xsl-region-body">
          <fo:block>
            <xsl:value-of select="Content"/>
          </fo:block>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
</xsl:stylesheet>

```

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

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



WebSphereDevelopersJournal.com

WebSphere
DEVELOPER'S JOURNAL



Introductory
Charter Subscription

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

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

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

It's that easy

The
World's
Leading
Independent
WebSphere
Developer
Resource



Using stylesheets to separate content from presentation

An Introduction to XSLT

Part 7

You've probably heard the propaganda by now: XML blesses you with a way to separate content from presentation. Separation in turn yields productive gains over HTML and other data formats used to manage content.

In a process sometimes called *single sourcing*, the content of an XML document can be formatted for display in a Web browser, reformatted for delivery to such devices as mobile phones and handheld computers, and converted into a PDF file suitable for printing.

A news story, for example, that has been marked up in XML can be transformed into HTML for publication on a Web site, converted into Wireless Markup Language (WML) for display on a cell phone's screen, and rendered into a format from which a PDF file can be generated. Meantime, parts of the news story, such as paragraphs that supply background information about a given subject, can be reused in other XML-encoded news stories. What makes all this possible? XSLT.

Extensible Stylesheet Language for Transformations, or XSLT, is a functional programming language that enables us to bridge the gap between content and presentation by providing a means of specifying how a content-based XML document is transformed into a presentation-oriented document or data format.

This tutorial, written primarily for content authors, technical writers, Web designers, and other nonprogrammers, is the first in a short series that aims to introduce you to the basics of XSLT and help you get started using it to transform XML documents into HTML. The focus is not on data-centric documents but on such narrative-centric documents as reference manuals, help files,

essays, stories, instructions, books, and magazine articles.

I'll begin by briefly summarizing the background of XSL and contrasting XSLT with XSL-FO and Cascading Style Sheets (CSS). Along the way I'll discuss what XSLT is, what it can do, and how to use it in conjunction with CSS to separate content from presentation. The next section will introduce you to XSLT's most fundamental component, the template rule, and demonstrate how to write one, culminating in a simple XSLT stylesheet that uses Internet Explorer 5.5's XSLT processor to transform a simple XML document into an HTML document.

XSLT, XSL-FO, and CSS

The birth of XSLT breaks down like this: Extensible Stylesheet Language, or XSL, was conceived as an XML application for expressing stylesheets capable of manipulating XML documents. After its submission to the W3C in 1997, XSL split into two stylesheet standards: Extensible Stylesheet Language for Transformations (XSLT) and Extensible

Stylesheet Language Formatting Objects (XSL-FO).

XSLT is a language for transforming XML documents into typically other XML documents, HTML, and WML, but can also be used to produce documents in plain text and XSL-FO.

XSL-FO is used to describe the layout of XML documents for printing, with the end product usually being in Portable Document Format (PDF). XSL-FO, a complex standard that some refer to simply as XSL, may eventually also be employed to lay out Web pages, though this use is unsupported by current versions of Web browsers. You can find more information about XSLT and XSL-FO, including their specifications, on the World Wide Web Consortium's Web site, www.w3c.org. This column focuses on XSLT; later tutorials will expand on XSLT and delve into XSL-FO.

The early XSL proposals also gave rise to another breakaway standard, XPath. It grew out of the location-finding aspects of the XSL and XPointer specifications, which had been independently using similar mechanisms to find information in XML documents. XSLT uses XPath heavily to locate specific nodes or node sets within XML documents. Details about XPath can be found at www.w3c.org/TR/xpath.

XSL has another related standard, though, unlike XSL, it isn't based on XML: Cascading Style Sheets. In the context of Web publishing, CSS can also be used, at least theoretically, to statically format XML documents for display, but it can't be used to transform them in any meaningful way. For



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.

example, XMetal, a tool for writing and editing in XML, uses CSS to format XML documents for display in its normal view. But until there is much stronger Web browser support for displaying XML documents with CSS, it isn't a practical option for Web publishing.

CSS isn't a competitor of XSLT. It's a method for statically formatting an XML document in a way that allows you to separate formatting and styling information from the document, but it doesn't allow you to dynamically transform an XML document into another data format. With CSS you can't manipulate the structure of an XML document, change the ordering of content, or dynamically generate a table of contents from a set of headings. Only XSLT can do that.

Using CSS to complement XSLT, however, is a powerful strategy for building Web pages – a strategy that splits presentation into what I call formatting and styling. *Formatting* can 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. Although the distinction between the two is not always clear-cut, formatting typically appears in the form of elements, styling information in the form of attribute-

value pairs. For example, in `<h1 style="color: olive;">` the h1 element formats the text as a first-level heading while the values of the style attribute, used for specifying inline CSS styles, reflect how the text should be styled.

You can gain a great deal of utility from separating as much styling information as possible from the formatting and placing it in a Cascading Style Sheet, although until the second major version of CSS (CSS Level 2) is better supported by browsers, your HTML formatting markup will still have to retain some styling information, such as that for tables.

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 changes in one file, the Cascading Style Sheet. *Cascading Style Sheets: The Definitive Guide*, published by O'Reilly, provides a detailed account of how to use CSS. The W3C CSS specifications are available at www.w3c.org/Style/CSS/. In this tutorial, I'll begin demonstrating how to use CSS with XSLT to separate styling from formatting, and I'll expand on the topic in later tutorials.

Preliminaries: Principles of Separation

Other principles of separation can be immensely useful in building well-engineered, text-based Web pages with XML, XSLT, CSS, and HTML. An overarching objective is to use XML to structure content and XSLT and CSS to format it in a way that minimizes redundancy and maximizes flexibility, including the capability to repurpose content and publish it in various formats. Most of the principles that follow are best applied to narrative-oriented documents that will be published as white papers, technical manuals, help files, essays, and so forth. Keep in mind that these principles are merely a guide and that the list of exceptions is long: your data, purpose, audience, delivery format, and other factors will influence how you engineer your system's own matrix of structure, metainformation, parameters, content, and presentation.

- **XML documents:** Strive to maximize the content that exists as text, not as tags, in your XML documents. In other words, content that you expect end users to see should usually be set as content, not as elements or attributes. Avoid setting content as tags, either as elements or attributes, even

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





if doing so comes at the cost of some redundancy; it's easier for content authors to work with content that is out in the open, not hidden as the value of attributes or as elements themselves. Instead, use elements to describe the structure or content of your material; use attributes to capture meta-information about the content or its structure and to encode parameters that are used to process the content. Place recurring content, especially content that may change, such as the name of a product under development, in entities. Shy away from placing formatting or styling information in your XML documents, although it may be expedient to include some table- and image-formatting instructions.

- **XSLT stylesheets:** Place as much formatting information as possible in your XSLT stylesheet and eschew using it as a container for standard content. Instead, place content out in the open in your XML documents even if doing so comes at the expense of a little duplication. However, highly variable content, such as the date of publication and the version number, may be best placed in the XSLT stylesheet as entities. It may also benefit you to make exceptions for content that varies by audience, as in the case of parameterized settings used in internationalization.
- **Cascading Style Sheets:** As I mentioned above, these should complement the XSLT stylesheet by containing as many as possible of the formatting code's visual styling properties.

It may benefit you to make exceptions for content that varies by audience

In the examples that accompany this and later tutorials in the series, you'll see some of these principles at work.

XSLT's Atom: 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. (Actually, it's the XSLT processor that takes an XML document, called a *source tree*, and an XSLT stylesheet as input and uses them to produce a result tree as output, which can then be serialized into a file, but we don't need to worry about such technicalities just yet.) A template rule is an XSL element that matches a node in the XML source document and typi-

cally applies an output format to it. For example, say you have the following simple XML document:

```
<?xml version="1.0"?>
<source>
  <message>Greetings and
    Salutations.</message>
</source>
```

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

```
<xsl:template match="/">
  <html>
    <body>
      <h1><xsl:apply-
        templates/></h1>
    </body>
  </html>
</xsl:template>
```

In the above 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="source">`. 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.

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

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



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

only one child of the <source> node in our XML file, <xsl:apply-templates> suffices to print the meager contents of the file. If, however, the <source> 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 processed and outputted.

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:

1. We need to add an XML processing instruction to the top of the stylesheet.
2. 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/> for details):

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl=
  "http://www.w3.org/1999/XSL/
  Transform"
  version="1.0">
  <xsl:template match="/">
    <html>
      <body>
        <h1><xsl:apply-
          templates/
        ></h1>
      </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 IE 5.5 or greater, 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, which in this case is located in the same directory as the source file, as the path in the href attribute's value testifies:

```
<?xml version="1.0"?>
<?xsl:stylesheet type="text/xsl"
  href="my_stylesheet.xsl"?>
<source>
  <message>Greetings and
  Salutations.</message>
</source>
```

Let's expand our minimal stylesheet to do a few more things. First, we'll modify our template rule to output the message element specifically (as opposed to all the children of the root element); second, we'll add a link to a CSS 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="source">
    <html>
      <head>
        <link rel=
          "stylesheet" type=
          "text/css" href=
          "my_styles.css"/>
      </head>
      <body>
        <h1><xsl:value-
          of select=
          "message"/></h1>
      </body>
```

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

This stylesheet first matches the root node explicitly by name (source), 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 following code:

```
h1 { color: olive; }
```

This simple code selects the <h1> element and renders it in olive in Internet Explorer, effectively separating the color aspect of the heading's visual style from its HTML formatting code in the XSLT stylesheet.

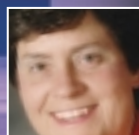
Because XSLT is a programming language, there's a great deal more to it than the simple examples shown above. To reinforce this column's brief introduction, I suggest you read Chapter 6, "Transformation: Repurposing Documents," in O'Reilly's *Learning XML*. I also suggest you start playing around with XSLT a little on your own; it's the best way to learn how to use it. Try creating an XML document and write some simple template matching rules like those above to process your source and output it in Internet Explorer. My next column will delve deeper into XSLT to unearth its complexity and accompanying power. ☉

SHOENISH @ RCN.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



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

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 23**

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



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

Prices subject to change.
Not responsible for typographical errors.

Attention Software Vendors:

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

SYS-CON MEDIA

XML-Journal: The Complete Library



JDJStore.com **\$53⁹⁹**

SYS-CON MEDIA

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



JDJStore.com **\$229⁹⁹**

SYS-CON MEDIA

CFDJ: The Complete Library



JDJStore.com **\$71⁹⁹**

MACROMEDIA

ColdFusion Server 5

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**

SYBASE

Adaptive Server Enterprise for WINNT v12.0

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



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

POLYCOM

VoiceStation Personal Teleconferencer

Designed to improve your business communications while increasing productivity, VoiceStation 100 enables you to conduct remote meetings that are as natural as speaking face-to-face.

Features: full-duplex audio performance featuring Polycom's Acoustic Clarity Technology™, dynamically adjusts to any small room's acoustic environment, 3 microphones w/ 360° room coverage, high-quality speaker, easy to install / easy to use, redial, flash, hold & mute, data port (110V versions only), RCA output jack.



VoiceStation Personal Teleconferencer **\$259⁹⁹**

CASIO

EM500 Multimedia Cassiopeia

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

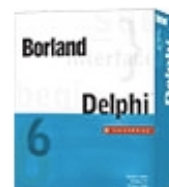


EM500 Multimedia Cassiopeia **\$428⁹⁹**

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 2000

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



RoboHELP Office 2000 **\$898⁹⁹**

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**

JASC

Quickview Plus v6.0

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



Quickview Plus v6.0 **\$45⁹⁹**

SMC

Wireless 11 Mbps Access Point with PC Card

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



Wireless 11 MBPS Access Point **\$339⁹⁹**

SITRAKA

JClass Enterprise Suite v 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**

LINKSYS

WAP11 Wireless Network Access Point

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



WAP11 Wireless Network Access Point **\$174⁹⁹**

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**

ARCHOS

Portable 6GB MP3 Player/USB

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



Portable 6GB MP3 Player/USB **\$268⁹⁹**

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**

POINTBASE

Mobile Edition 3.5

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



PointBase Mobile Edition 3.5 **\$194⁹⁹**

MACROMEDIA[®]

JRun[™] Server 3.0/3.1 Enterprise 2 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 (2 CPU Licenses) **\$8602⁹⁹**

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**



BRIDGING THE GAP BETWEEN JAVA

JAVA AND XML ARE PERFECTLY MARRIED. JAVA REPRESENTS A TECHNOLOGY EVOLUTION FOR PLATFORM-INDEPENDENT DEVELOPMENT AND DEPLOYMENT, AND AN EFFECTIVE MECHANISM FOR ACHIEVING DISTRIBUTED COMPUTING. XML, A VERY SIMPLE CONCEPT, HAS TAKEN THE INDUSTRY BY STORM AND IS REVOLUTIONIZING HOW DATA IS REPRESENTED AND EXCHANGED WITHIN A COMPANY AND BETWEEN ENTERPRISES. IN A NUTSHELL, JAVA REPRESENTS PORTABLE CODE AND XML REPRESENTS PORTABLE DATA – A PERFECT MARRIAGE.

IN AN EFFORT TO FUEL THIS MARRIAGE, SUN HAS LAUNCHED A SET OF TECHNOLOGIES COLLECTIVELY KNOWN AS THE JAX PACK. JAX PACK – ESSENTIALLY A BUNDLED SET OF JAVA TECHNOLOGIES FOR XML – CONSISTS OF JAXP (XML PROCESSING), JAXB (XML BINDING),

JAXM (XML MESSAGING), JAX-RPC (XML-BASED RPC), AND JAXR (XML REGISTRIES).

THESE TECHNOLOGIES FOR XML ARE AVAILABLE AS SEPARATE SPECIFICATIONS, APIS, AND REFERENCE IMPLEMENTATIONS (SOME SPECS/IMPLEMENTATIONS ARE GENERALLY AVAILABLE; OTHERS ARE WORKS IN PROGRESS). HOWEVER, JAX PACK IS ALSO AVAILABLE AS A COMBINED SET OF ALL THE JAX TECHNOLOGIES IN A SINGLE DOWNLOAD. THE STANDARDS/SPECIFICATIONS REPRESENTING THE VARIOUS JAVA TECHNOLOGIES FOR XML HAVE BEEN DEVELOPED COLLABORATIVELY WITH THE JAVA COMMUNITY PROCESS (JCP; SEE SIDEBAR).

THE OBJECTIVE OF THIS ARTICLE IS TO WALK THROUGH THESE APIS, REVIEW THEIR FUNCTIONALITY, AND, USING CODE EXAMPLES, ILLUSTRATE HOW THEY CAN BE USED WITHIN APPLICATIONS.

JAXP

AND XML TECHNOLOGIES



WRITTEN BY HITESH SETH

JAXP

Fundamental to the fusion of Java and XML technologies, the Java API for XML Processing provides a set of APIs for parsing, creating, and transforming XML documents. JAXP supports both the memory-based DOM2 (Document Object Model) and the event-based SAX2 APIs for XML parsing (Simple API for XML). With the 1.1 release JAXP also supports XSLT-based transformations, a critical component that adds a new level of abstraction and ease to transforming XML documents into other XML vocabularies, plaintext, and even print media (using XSL Formatting Objects).

Given this introduction to JAXP, you may wonder why, since a large number of companies have built their own XML parsers and XSLT processors, we need another one.

The answer is simple: JAXP isn't just another XML parser. Instead, what JAXP APIs and the reference implementation provide is a standard set of Java APIs that define a factory-based pluggable framework for XML parsers and XSLT processors (as illustrated in Figure 1).

The reference implementation of JAXP uses the newly released Apache Crimson ([http://xml.](http://xml.apache.org/crimson/)

<http://xml.apache.org/crimson/>) as the XML parser and Apache Xalan (<http://xml.apache.org/xalan-j/index.html>) as the default XSLT processor, but it's really a matter of calling some APIs and setting some system properties if you'd like to use your favorite parser/processor.

A key component missing from the JAXP API set is the support for XML Schemas, expected in future releases. JAXP does support document type definition (DTD) based XML validation and using a third-party parser such as Apache Xerces (<http://xml.apache.org/xerces-j/index.html>) or Apache Xerces2 (<http://xml.apache.org/xerces2-j/index.html>). Setting a couple of attributes to indicate that your application needs XML Schema validation is what's required.

JAXP has been a popular API; it's been incorporated in the newly released J2EE v1.3 specification to provide XML parsing/processing capability to server-based enterprise applications. It's also been incorporated by a large number of third-party application servers and other products. As another endorsement of this key API, JAXP 1.1 has been included as a core aspect of the new version of the Java platform (J2SE v1.4).

A simple code snippet is worth a thousand words. Following is a sample XML document (Order.xml) defined by a DTD – Order.dtd. I'll show how this document can be processed and transformed using the JAXP API set, and will use the same example in later sections to demonstrate how the various Java technologies for XML work together.

JAVA COMMUNITY PROCESS

Java Community Process, or JCP, is an open organization effort in which experts from ISVs and developers of Java technology participate jointly as required by their charter in reviewing existing Java technologies and introducing new features to the Java technology set.

Nearly all of the leading Java-related standards – Java 2, Enterprise Edition (J2EE), Java 2, Micro Edition (J2ME), Java APIs for XML (JAXP, JAXB, the focus of this article), and the latest versions of Java 2, Standard Edition (J2SE) – have been a result of the collaborative effort of JCP participants. Originally established by Sun Microsystems, JCP has been in existence since 1995.

JCP is split into multiple expert groups, each working toward a development/enhancement of a Java technology; the expert group is known as JSR (Java Specification Request).



```
Id="+root.getAttribute("id");
NodeList nl = root.getChildNodes();
for (int i=0;i<nl.getLength();i++) {
    ... // Process other nodes
}
```

SAX and DOM provide the basic low-level generic API to parse and validate an XML document. Earlier we introduced XSLT, an XML standard that can be used to transform an XML document/stream into another XML, HTML, WML, or any other format.

Let's use the other key component of the JAXP API, stylesheets, to transform an XML document. Note the abstraction and ease of use XSLT provides us. The following example uses XSLT to create HTML on the fly as a Java application, but this same functionality can also be used to create a full-fledged content management application using J2EE APIs such as Java servlets and JavaServer Pages.

```
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
...
TransformerFactory tFactory =
    TransformerFactory.newInstance();
Transformer transformer = tFactory.newTransformer(
    new StreamSource("Order.xml"));
transformer.transform(new StreamSource("Order.xml"),
    new StreamResult(System.out));
...
```

Transformation of XML documents using XSLT requires the creation of an XSLT-based stylesheet. Following is the simple stylesheet that converts the Order.xml into HTML:

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

```
<?xml version="1.0"?>
<order id="5677">
  <date>11-12-2001</date>
  <company>Silverline Technologies</company>
</order>
```

Following is the XML structure – Order.dtd:

```
<!ELEMENT order (date, company)>
<!ATTLIST order
  id CDATA #REQUIRED>
<!ELEMENT date (#PCDATA)>
<!ELEMENT company (#PCDATA)>
```

Now let's use the JAXP API set to create a simple Java application to parse the XML document. In this scenario I'll use the DOM API to parse the various elements and attributes of the Order.xml structure:

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
...
DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
DocumentBuilder db =
dbf.newDocumentBuilder();
Document doc = db.parse(new
File("Order.xml"));
Element root = doc.getDocumentElement();
System.out.println("Order
```

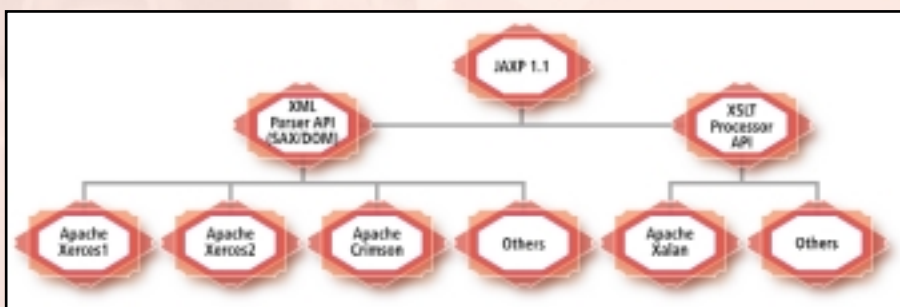


FIGURE 1 JAXP architecture



FIGURE 2 JAXB programming model

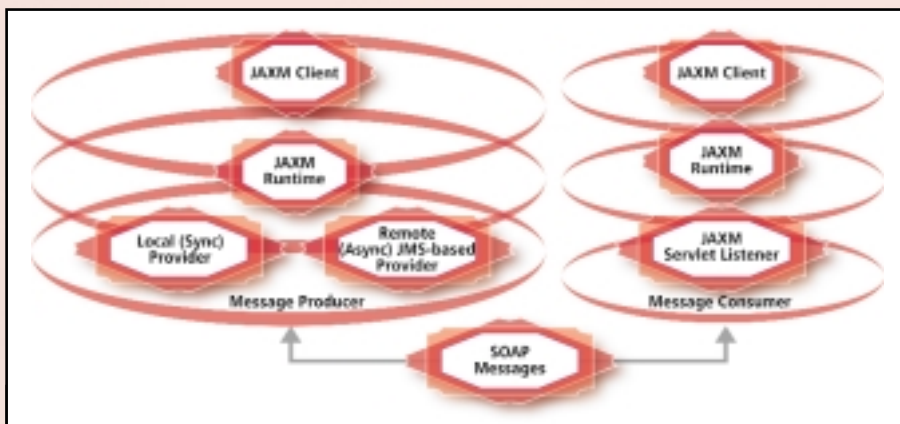


FIGURE 3 JAXM programming model

2002

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

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



JAVA, XML,
WEB SERVICES
AND .NET
TECHNOLOGIES

web **services** **EDGE**
 conference & expo

INTERNATIONAL WEB SERVICES CONFERENCE & EXPO

JDJ **EDGE**
 conference & expo

INTERNATIONAL JAVA DEVELOPER CONFERENCE & EXPO

XML **EDGE**
 conference & expo

INTERNATIONAL XML CONFERENCE & EXPO

Fundamentally Improving the Speed,
 Cost and Flexibility of Business Applications

Who Should Exhibit...

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

These Leading *i*-Technology Shows Feature...

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

Who Should Attend...

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

**ONLINE EARLY BIRD REGISTRATION
 OPENS FEBRUARY 15**

FOR MORE INFORMATION

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

TO EXHIBIT/SPONSOR

CALL 201-802-3004/201-802-3069

WWW.SYS-CON.COM

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

```

<xsl:template match="order">
  <html>
<body>
  <b>Order Id:</b><xsl:value-of select="@id"/><br/>
  <b>Date:</b><xsl:value-of select="date"/><br/>
  <b>Company:</b><xsl:value-of select="company"/><br/>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

JAXB

Whereas JAXP provides the basic/core API for processing XML, there are really two kinds of implementations for an XML parser that supports the JAXP APIs. On the one hand we have DOM, which provides a tree-based generic navigation for an XML document; on the other we have SAX, a fast, event-based, but very low level API. The two have often been compared on the basis of speed, resource requirements, and ease of use, but as illustrated previously, even by using the higher level DOM interface, the simplest Java application for processing the basic Order.xml document requires a certain amount of coding (we didn't really account for any error handling and processing logic).

In a nutshell, there's a need for a seamless interface that provides an easy-to-use programmatic interface for Java developers to parse, manipulate, and create XML documents.

JAXB – Java Architecture for XML Binding (previously known as Project Adelard) – fills this gap by providing performance-centric bidirectional mapping between Java objects and XML documents. Whereas a JAXB implementation may use JAXP-based XML core parsers underneath, when compared to DOM and SAX, JAXB truly provides an easy-to-use alternative mechanism for processing XML documents.

JAXB is divided into two components: a *schema compiler* and a *binding framework*. The XML-based binding schema is used to bind the XML DTD with Java-based artifacts such as the primitive and complex type mapping, customization of the generated classes (packages, class names, method names, constructors, etc.), and so on.

A schema compiler, illustrated in Figure 2, then creates a set of predefined Java classes that build a Java object tree corresponding to the validated XML document supported by the mapped schema rules. Instances of the Java classes, which use the JAXB runtime, can then be used to process and manipulate XML content from within the known Java environment. Even though JAXB-based Java classes do store the resulting mapped XML in memory, the memory overhead typically caused by the generic XML parser is significantly reduced, with the added benefit that you can strictly validate XML documents.

A current limitation of the JAXB specification is that it addresses only DTD-based XML definitions. We can expect future releases to provide an

implementation on top of the prevalent XML metadata standards, such as XML Schema.

JAXB Example Scenario

Using a simple scenario, let's explore the JAXB API and its benefit for binding XML documents to Java objects. First we need to create an XML Java Binding Specification file to map the various datatypes with the DTD. We'll again build on top of the Order example used previously (Order.xjs):

```

<?xml version="1.0"?>
<xml-java-binding-schema version="1.0ea">
  <element name="order" type="class" root="true">
    <attribute name="id" convert="int"/>
    <content>
      <element-ref name="date"/>
      <element-ref name="company"/>
    </content>
  </element>
  <element name="company" type="value"/>
  <element name="date" type="value"/>
</xml-java-binding-schema>

```

Executing the XML Java schema compiler, xjc (java com.sun.tools.xjc.Main Order.dtd Order.xjs), creates Order.java. This class file can be used by any Java application to bind XML documents to Java objects. Following is the example application (ReadOrder.java):

```

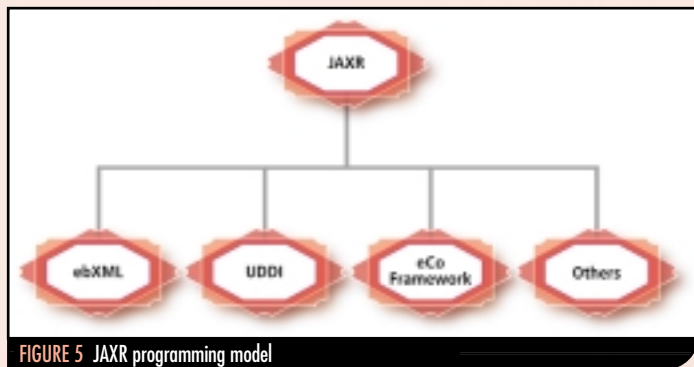
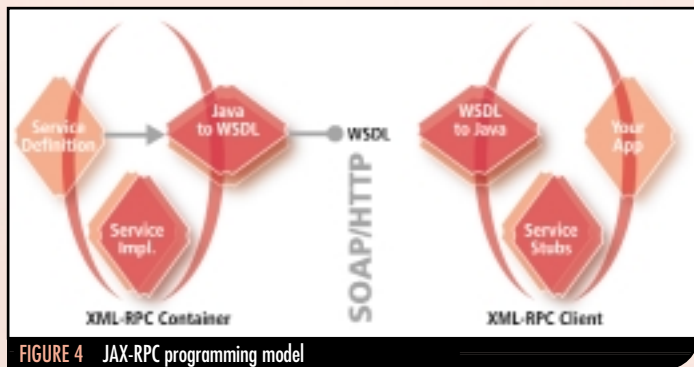
import javax.xml.bind.*;
import javax.xml.marshall.*;
...
Dispatcher d = Order.newDispatcher();
File orderFile = new File("Order.xml");
FileInputStream fis = new FileInputStream(orderFile);
Order order = (Order) (d.unmarshal(fis));
System.out.println("Order Id: "+order.getId());
System.out.println("Company: "+order.getCompany());
System.out.println("Date: "+order.getDate());
...

```

Running the Read Order Application would result in printing the order ID, company name, and order date.

JAXM

Java API for XML Messaging (JAXM, earlier known as the M Project), as the name suggests, provides a lightweight API for XML-based messaging based on SOAP 1.1 (including attachments support). The advent of the



Plan to Exhibit

Provide the Resources To
Implement Wireless Strategy

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

INTERNATIONAL

WIRELESS BUSINESS & TECHNOLOGY

CONFERENCE & EXPO

C O N F E R E N C E & E X P O

C O N F E R E N C E & E X P O



Shaping Wireless Strategy for the Enterprise

Santa Clara, CA

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

Plan to Attend the 3-DAY Conference

May 7-9, 2002

WHO SHOULD ATTEND

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

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

Conference Tracks

Track One: Development

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

Track Two: Connectivity

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

Track Three: Wireless Apps

Education
Health Care
Entertainment
Transport
Financial Services
Supply Chain
Management

Track Four: Hardware

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

Track Five: Business Futures

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

**FOR EXHIBIT & SPONSORSHIP
INFORMATION PLEASE CALL**

201 802-3004

SPEAKER PROPOSALS INVITED

WWW.SYS-CON.COM

**SHAPE
YOUR
WIRELESS
STRATEGY...
SAVE THE
DATES!**



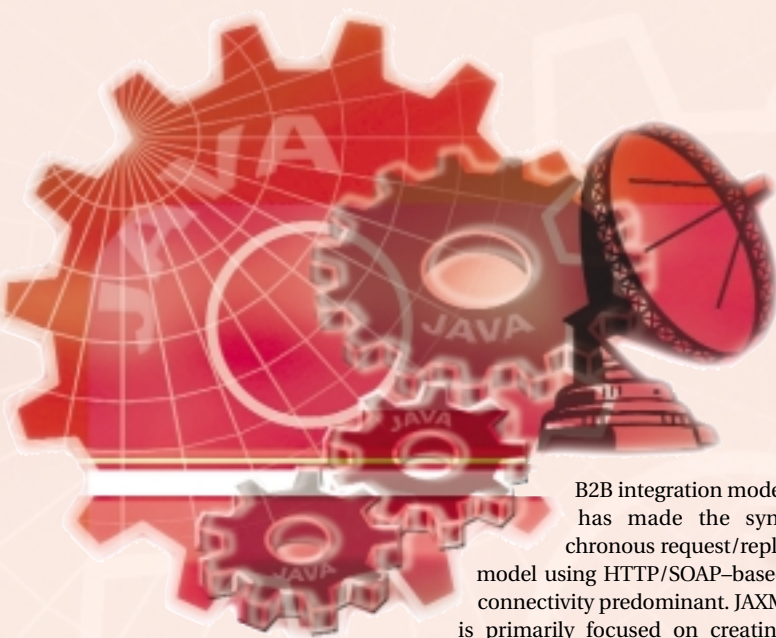
EXCLUSIVE SPONSORSHIPS AVAILABLE

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

**SYS-CON
MEDIA**

**SYS-CON
EVENTS**

WWW.WIRELESSEGE2002.COM



B2B integration model has made the synchronous request/reply model using HTTP/SOAP-based connectivity predominant. JAXM is primarily focused on creating and sending/receiving SOAP-based messages.

As a basic API, JAXM is pretty lightweight in messaging, but allows flexibility for implementation to JAXM providers through the concept of profiles.

A highlight of the current release of JAXM and the reference implementation is the ebXML profile, which supports the asynchronous messaging model based on the ebXML Messaging Specification. This model is critical in high-volume and loosely coupled scenarios, in which as a Web services client you'd like a "fire and forget" messaging scenario – send the message through a messaging layer and let the consumers of the message worry about when to process it.

Apart from the loose-coupling benefit, a key aspect of the asynchronous messaging models is the inherent service-level guarantees you can achieve. For instance, achieving features such as guaranteed delivery in a typical request/response interaction can be a daunting task, one that typically requires an implementation to implement substantial error-handling functionality. On the other hand, guaranteed delivery is one of the highlights of a loosely coupled messaging model.

Asynchronous communication isn't new to the IT industry; messaging systems such as MQSeries, TIBCO, and MSMQ have been the foundation for integrating disparate systems for a large number of organizations. Java Message Service (JMS), another specification from Sun Microsystems, takes a messaging vendor-independent, provider-based approach (similar to JDBC) to provide a baseline API for messaging systems. JAXM takes this one step further; using a concept of profiles, it bridges the interaction of standards-based Web services with the JMS-based pluggable messaging infrastructure.

JAXM defines the interactions between two entities, a JAXM client and a JAXM provider. JAXM clients use the API to create/send messages, whereas JAXM providers implement the API to support the actual transmission and reception of SOAP messages, as illustrated in Figure 3.

To uncover the details of how the JAXM API works, I'll create a simple JAXM client that calls a SOAP service that an external vendor/customer can use to get the current order status for a particular order (see Listing 1).

JAX-RPC

Remote Procedure Call (RPC), a mechanism that enables a remote machine (typically the server) to call a procedure, isn't new to the Java platform. Remote Method Invocation (RMI) has been a fundamental component of the Java platform and the basis of key enterprise Java technologies such as Enterprise JavaBeans (EJB). CORBA, another step toward the evolution of RPC, has provided a platform- and programming language-independent mechanism for calling remote procedures. Various implementations of RPC, including RMI and CORBA, have been key components in the design and development of distributed Java technologies.

It's always been a challenge when the "remote server" is located in another company; this is known as being outside the firewall. Even though there are vendor solutions for tunneling RPC calls within ubiquitous Internet communication standards such as HTTP/HTTPS, setting up a distributed, loosely coupled B2B architecture between enterprises has always been challenging. This has led to the development of standards such as SOAP, a specification that defines an XML-based protocol for representing loosely coupled distributed remote procedure calls. W3C has recognized SOAP and has initiated a working group for the development of the next generation of XML-based messaging, XML Protocol (XMLP).

Continuing with the philosophy of standards-based access, the Java API for XML-based RPC (JAX-RPC) defines a set of APIs for facilitating XML-based RPC communication with the Java platform. The initial release of JAX-RPC (1.0), currently in community draft stage, requires implementations to support SOAP 1.1 on top of HTTP 1.1 communications. However, the whole JAX-RPC initiative has paved the way for supporting other protocols such as XMLP and communication mechanisms such as HTTPS. Fundamental to JAX-RPC is the type mapping between Java and XML datatypes and a programming model. The JAX-RPC runtime can be implemented as a client component, building on top of J2SE, or an enterprise/server-based component leveraging either a Servlets 2.3-based model or the fullblown capability of J2EE 1.3-based application servers.

Web Service Description Language (WSDL) has been used by JAX-RPC as the standard for specifying JAX-RPC-based services; the specification defines a bidirectional mapping between WSDL and Java. Implementations can support multiple interaction modes, including synchronous request/response, one-way RPC, and nonblocking RPC invocation.

API	Related XML Technologies & Standards	Function	Java Packages	Current Version	Specification Status	Reference Implementation Status	JCP Expert Group
JAXP	<ul style="list-style-type: none"> •SAX/SAX2 •DOM/DOM2 •XSLT •XPath 	XML Parsing & XSLT Processing	<ul style="list-style-type: none"> •javax.xml.parsers •javax.xml.transform 	1.1.3	Final release (02/6/2001)	General release available (Part of fall '01 JAX Pack release)	JSR-63
JAXB	<ul style="list-style-type: none"> •DTD 	XML Binding	<ul style="list-style-type: none"> •javax.xml.marshal •javax.xml.marshal •javax.xml.schema 	0.21	Working draft (05/30/2001)	Early access implementation	JSR-31
JAXM	<ul style="list-style-type: none"> •SOAP •ebXML Messaging Specification •XML Schema 	XML/SOAP-based messaging	<ul style="list-style-type: none"> •javax.xml.messaging •javax.xml.soap 	1.0	Final release (10/30/2001)	1.0 Reference implementation available (Part of fall '01 JAX Pack release)	JSR-67
JAX-RPC	<ul style="list-style-type: none"> •SOAP •W3C XP •WSDL 	XML-based RPC	<ul style="list-style-type: none"> •javax.xml.rpc •javax.xml.soap 	0.5	Community draft (10/10/2001)	Not yet available	JSR-101
JAXR	<ul style="list-style-type: none"> •UDDI •ebXML 	XML Registries	<ul style="list-style-type: none"> •javax.xml.registry 	0.6	Public review draft (07/30/2001)	Not yet available	JSR-93

TABLE 1 Java technologies for XML

To understand how JAX-RPC works, let's look at the various steps involved in developing, deploying, and using an XML-RPC-based service. Developers familiar with the Java RMI/CORBA programming model would find this very similar, as illustrated in Figure 4.

- **Service definition:** Define the service using the familiar Java RMI definition syntax:

```
public interface OrderStatusProvider extends java.rmi.Remote {  
    public String getOrderStatus(String orderId)  
        throws java.rmi.RemoteException;  
}
```

- **Service implementation:** Implement the service. Depending on the type of JAX-RPC container implementation, the service can be implemented through a J2SE container (by extending the PortableRemoteObject class) or by using a J2EE component model such as session EJB.
- **Service deployment:** Deploy the service on a server-side JAX-RPC-based container. This step also creates and activates the associated bindings such as SOAP/HTTP, depending on the protocols supported and the deployment configuration.
- **The deployment tool:** This tool exports the service definition as WSDL.
- **Service invocation:** From a client's perspective, a client uses a WSDL-to-Java compiler to generate the necessary stubs to invoke the service.
- The stub can then be used by client programs.
- Alternatively, the client may also use a Dynamic Invocation Interface (DII) to bind to the service without the stub generation process.

JAXR

Java API for XML Registries is a standard programming API for interfacing with business registries (think of a registry as a bulletin board, a set of yellow pages for Web services) of Web services providers. These registries can be owned privately by an organization or can be a collaborative exchange-based effort fueled by a specific vertical group (such as a chemical or automotive exchange). An important component in the evolution of Web services, we've already seen efforts such as UDDI (Universal Description, Discovery, and Integration), OASIS, eCo Framework, and ebXML lead the development of such registries. JAXR is an attempt to provide a common abstraction for multiple registry standards.

JAXR defines the concept of a capability profile, which allows registry providers to classify themselves based on the level of support. Currently two profiles – Level 0 Profile and Level 1 Profile – have been defined. As the nomenclature probably suggests, Level 0 is a baseline profile that defines a business-level API; registry providers of a Level 1 Profile can optionally support a generic API for flexibility and advanced registry capabilities.

JAXR builds on top of other JAX technologies, leveraging JAXP/JAXB for processing XML, JAX-RPC for communication, and JAXM for XML-based messaging. At the time this article was written, JAXR was in the early phase of its development, with only a draft version of the specification available. The specification did, however, define how JAXR APIs map to ebXML and UDDI. Based on the specification, the code snippet in Listing 2 shows a possible use of the JAXR APIs – to connect with an internal UDDI registry service.

From a business perspective, to understand how JAXR and the overall business registry concept works, consider a business scenario in which a computer manufacturing company, A, wants to order some raw materials. A queries the public UDDI registry, hosted by a vertical exchange, C, and through that is able to locate a wholesale supplier, B, that provides the appropriate raw materials. Through well-defined services, an integration workflow is then able to call an order entry system of that supplier, order goods.

The whole B2B interaction can happen in an automated, seamless fashion. JAXR will be a significant component in implementing the scenario in which JAXR APIs will be used by A to query the registry and possibly by exchange with C to implement the registry itself. One reason why JAXR would be significant is that multiple registry standards are evolving and, through various factors such as evolution, market pressure, and customer requirements, companies and vendors would potentially be required to support multiple registry formats and exchanges (see Figure 5).

Availability

Table 1 summarizes the availability and high points of the different Java technologies for XML. At the time of this writing, two technologies, JAXP 1.1.3 and JAXM 1.0, were available as part of the fall 2001 JAX Pack. An early-access implementation of JAXB was also available. However, JAX-RPC and JAXR are still developing technologies and reference implementations were not available.

Conclusion

I think it's safe to assume that Java technology is here to stay. Portable code executing on multiple platforms has turned out to be a great value proposition, at least for the server. An important participant in server-based interactions, specifically with B2B relationships, is XML. The JAX Pack provides a good combination of a wide range of Java technologies that would largely simplify developing applications around XML, whether it's parsing XML, transforming XML, writing Web services, or communicating synchronously/asynchronously using SOAP.

Also, as we've seen, most of these APIs are written using a driver/provider philosophy that makes the API more generic than the individual protocols themselves. As seen in Table 1, a lot of progress has occurred in the Java community to support the various XML technologies as part of the Java platform; some work still needs to be done – reference implementations need to be generally available, with vendor-specific enhancements and suites soon to follow. In a number of initiatives, we've already seen JAXP included as a core API; it's part of the J2EE 1.3 and J2SE 1.4 specifications. Overall, we can expect the JAX Pack to establish a set of baseline technologies for bridging the gap between the various components of the Java platform and the XML-based standards, thus building on the destined marriage between Java and XML technologies.

References

Following are some Web sites for you to look at while implementing and developing these technologies:

- *Java Community Process (JCP):* www.jcp.org
- *Java technology for XML:* <http://java.sun.com/xml>
- *Java XML Pack:* <http://java.sun.com/xml/javaxmlpack.html>
- *Java API for XML Parsing:* <http://java.sun.com/xml/jaxp/index.html>
- *Java Architecture for XML Binding:* <http://java.sun.com/xml/jaxb/index.html>
- *Java API for XML Messaging:* <http://java.sun.com/xml/jaxm/index.html>
- *Java API for XML-based RPC:* <http://java.sun.com/xml/jaxrpc/index.html>
- *Java API for XML Registries:* <http://java.sun.com/xml/jaxr/index.html>

AUTHOR BIO

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

H I T E S H . S E T H @ S I L V E R L I N E . C O M

LISTING 1

```

import javax.xml.messaging.*;

import javax.xml.soap.*;

...

SOAPConnectionFactory conf =
SOAPConnectionFactory.newInstance();

SOAPConnection con = conf.createConnection();

MessageFactory mf = MessageFactory.newInstance();

SOAPMessage msg = mf.createMessage();

SOAPPart sp = msg.getSOAPPart();

SOAPEnvelope envelope = sp.getEnvelope();

SOAPBody body = envelope.getBody();

SOAPBodyElement elm = body.addBodyElement(

envelope.createName("GetOrderStatus", "m",

"http://www.silverline.com/webservices/"));

elm.addChildElement(envelope.createName("orderId", "m",

"http://www.silverline.com/webservices/")).addTextNode("5677"

);

msg.saveChanges();

```

```

URLConnection urlEndpoint = new
URLConnection("http://server/mySOAPService");

SOAPMessage reply = con.call(msg, urlEndpoint);

...

```

LISTING 2

```

import javax.xml.registry.*;

...

RegistryClient client = new MyRegistryClient();

ConnectionFactory factory = (ConnectionFactory)

ctx.lookup("JAXRConnectionFactory");

Properties p = new Properties();

p.put("javax.xml.registry.factoryClass",

"com.sun.xml.registry.ConnectionFactory");

p.put("javax.xml.registry.queryManagerURL",

"http://silverline.com/uddi/inquiry");

p.put("javax.xml.registry.lifeCycleManagerURL",

"http://silverline.com/uddi/publish");

Connection connection = factory.createConnection(p, client);

Set credentials = new Set();

connection.setCredentials(credentials);

RegistryService service = connection.getRegistryService();

```

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

Your Own Magazine



SYS-CON
CUSTOM
MEDIA

Do you need to differentiate yourself from your competitors?

Do you need to get closer to your customers and top prospects?

Could your customer database stand a bit of improvement?

Could your company brand and product brands benefit from a higher profile?

Would you like to work more closely with your third-party marketing partners?

Or would you simply like to be a magazine publisher?

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

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

SYS-CON Custom Media can produce inserts, supplements, or full-scale turnkey print magazines for your company. Nothing beats your own print magazine for sheer impact on

your customers' desks... and a print publication can also drive new prospects and business to your Web site. Talk to us!

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

Contact us today!

EAST OF THE ROCKIES
Robyn Forma
robyn@sys-con.com
Tel: 201 802-3022

WEST OF THE ROCKIES
Roger Strukhoff
roger@sys-con.com
Tel: 925 244-9109

Premiering... this *January* subscribe Now!

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

FORFAST
DELIVERY

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

Organizations are being challenged to partner with other organizations in order to respond more rapidly to new business opportunities, increase the efficiency of business processes, and reduce the time-to-market for their products. To address these issues, they're typically required to develop interoperability between disparate legacy applications to support collaborative business processes.

This is accomplished by coordinating the exchange of business documents between applications in a predefined manner. For example, two insurance companies with different systems may need to exchange auto insurance claim data, such as a TIFF file for claims processing.

Enterprise applications that support these types of requirements are known as *business services*. Supporting business services through Web applications, commonly known as *Web services*, requires a careful evaluation of available technologies. An important underlying technology of Web services is Simple Object Access Protocol. SOAP enables applications to communicate with each other in a platform, language, and operating system-independent manner.

For those who need to use a Web service to perform a functionality such as sending a document in the form of attachments (e.g., a TIFF file) from one application to another using SOAP, a pertinent specification is the SOAP Messages with Attachments note. This article discusses this emerging W3C note and illustrates how it can be used with the Apache SOAP implementation.

SOAP MESSAGES

WITH ATTACHMENTS

SOAP Messages with Attachments Specification

As you know, SOAP is a simple, lightweight XML-based distributed computing protocol. The SOAP 1.1 specification essentially comprises three parts:

1. A framework for describing the contents of a SOAP message and how it's processed
2. An encoding standard for objects sent over SOAP
3. A mechanism for representing remote procedure calls (RPC) using SOAP.

A SOAP message is an XML document that comprises a SOAP envelope. Within the envelope is an optional SOAP header and a mandatory SOAP body. The SOAP message header represents the metadata of the message and provides a way to extend SOAP. The SOAP message body is the actual message payload. The details of a remote procedure call including the arguments are described in the envelope that is transported from one application to another over a selected protocol (e.g., HTTP). HTTP provides a firewall-friendly application-level protocol for communication between heterogeneous applications. The SOAP Messages with Attachments specification builds on SOAP 1.1.

Before we discuss this specification, we should briefly consider the concept of a Multipurpose Internet Mail Extensions multipart. MIME is leveraged in the SOAP with Attachments specification as well as in other related Web services specifications such as WSDL (Web Services Description Language). Typically, e-mail messages with attachments are sent over the Internet using Simple Mail Transfer Protocol and MIME. SMTP is limited to 7-bit ASCII text with a maximum line length of a thousand characters which results in the inability to send attachments. MIME addresses these limitations by specifying message header fields and allowing different related objects such as attachments to be included in the message body in the form of a MIME multipart.

For example, if the message body contains multiple independent objects of possibly different data types, the message body is divided into parts by boundaries. To indicate that the message body comprises a multipart structure of independent attachments, a Content-Type message header field is set. The message body boundary is defined with a parameter in the Content-Type field as shown below:

```
Content-Type: multipart/mixed;boundary="1995021309105517"
```

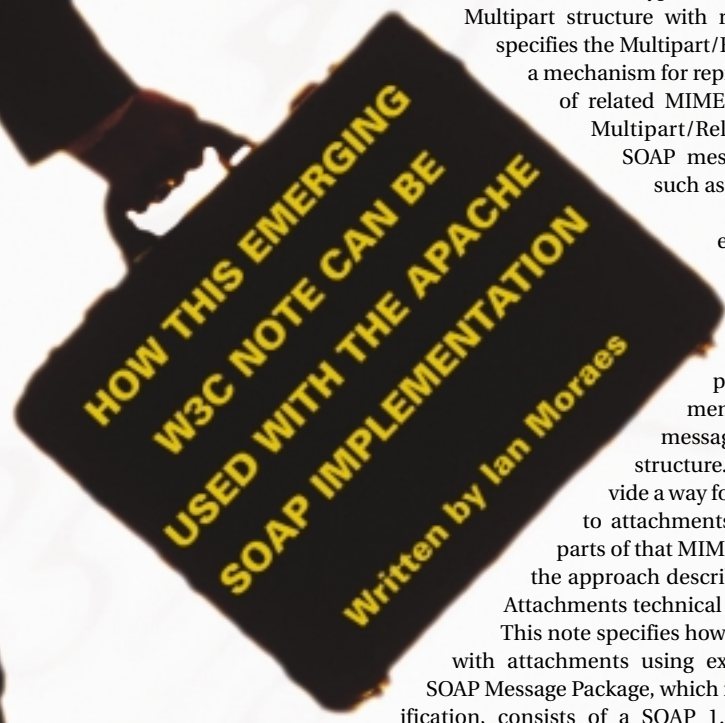
The Content-Type header can also be used to identify a Multipart structure with related MIME parts. RFC 2387 specifies the Multipart/Related Content-Type to provide a mechanism for representing an object that consists of related MIME body parts. As you'll see, the Multipart/Related Content-Type allows a SOAP message to reference attachments such as audio and image files.

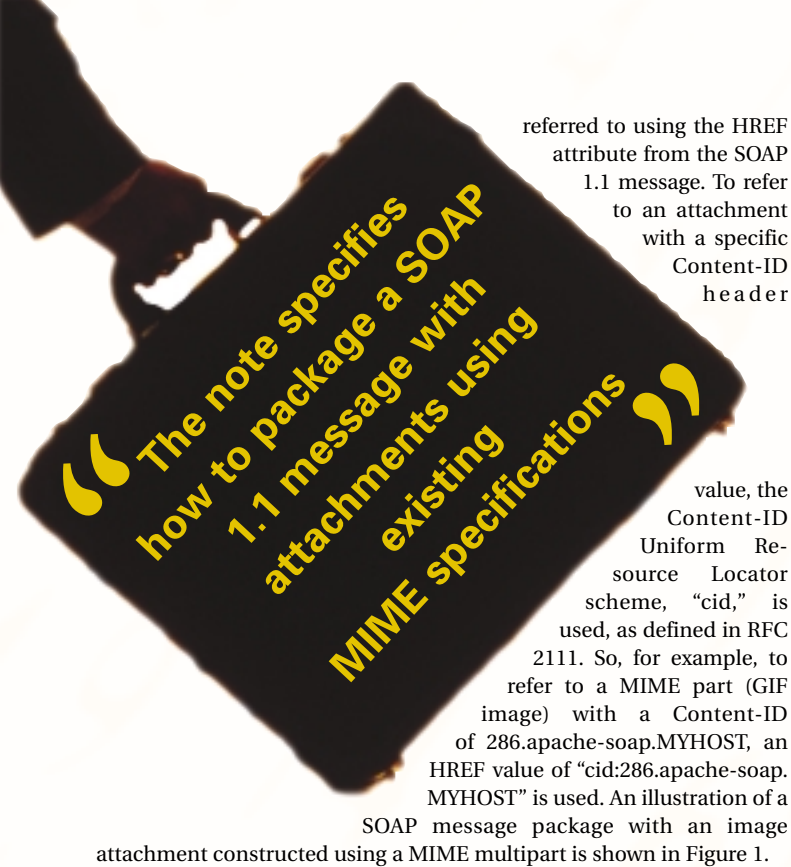
Armed with this basic knowledge of SOAP and MIME, let's discuss the concept of SOAP messages with attachments.

Let's assume we use the MIME specifications to support SOAP messages with attachments and we package a SOAP 1.1 message as a part of a MIME multipart structure. Further, let's assume we provide a way for that SOAP 1.1 message to refer to attachments that are represented as other parts of that MIME multipart. In a nutshell, this is the approach described in the SOAP Messages with Attachments technical note.

This note specifies how to package a SOAP 1.1 message with attachments using existing MIME specifications. A SOAP Message Package, which is a central concept in this specification, consists of a SOAP 1.1 message that can reference attachments, and any related attachments. A SOAP message package is built using the Multipart/Related Content-Type. The primary SOAP 1.1 message must be included in the root body part of the Multipart/Related container. Thus the type header of the Multipart container and the Content-Type header of the root body part will be the same (i.e., text/xml). The attachments in the Multipart container must be composed of a Content-ID MIME header or a Content-Location MIME header.

For the SOAP message to reference attachments in the MIME container, the SOAP 1.1 encoding rules are used. In the specification a URI that is a value of an HREF attribute can be used to refer to a resource. Since each attachment in the MIME container has a URI reference in its Content-Location or Content-ID header, it's possible for the attachments to be





referred to using the HREF attribute from the SOAP 1.1 message. To refer to an attachment with a specific Content-ID header

value, the Content-ID Uniform Resource Locator scheme, "cid," is used, as defined in RFC 2111. So, for example, to refer to a MIME part (GIF image) with a Content-ID of 286.apache-soap.MYHOST, an HREF value of "cid:286.apache-soap.MYHOST" is used. An illustration of a SOAP message package with an image attachment constructed using a MIME multipart is shown in Figure 1.

Apache SOAP Implementation of SOAP Messages with Attachments

The Apache Software Foundation (www.apache.org) provides an implementation of SOAP including support for SOAP Messages with Attachments. I've chosen to discuss the Apache implementation because it's freely available. Apache SOAP version 2.2, which can be downloaded from the Apache Web site, requires a number of components, such as a servlet container (e.g., Apache Tomcat), JavaMail, JavaBeans Activation Framework, and a JAXP-compatible XML parser. As you know, JavaMail is a J2EE API that provides a platform and protocol-independent interface for sending and retrieving e-mail messages. It interacts with the message content through another layer in the form of the JavaBeans Activation Framework. JAF provides a uniform way of determining the type of a message's content and encapsulating access to it. You can obtain more information on JavaMail and JAF from the **Resources** section.

Here's a quick overview of how a simple Web service can be constructed with Apache SOAP and Java. The first step is to define and implement a Web service. When developing a simple Java application to support a Web service using SOAP, you'll notice that there's no SOAP-specific code in your Java class because Apache SOAP manages the details pertaining to SOAP. After implementing the service, you deploy the service describing the operation(s) that it supports and assign the service a unique service ID. When a SOAP client uses the service by calling one of its operations, the Apache SOAP rpcrouter servlet responds to the request. The request is routed to the pertinent Java object that supports the Web service. The service object interfaces with the core business logic to perform the business function and a response is returned to the client.

After deploying a Web service, you could create a Java client that calls the Web service. The client program uses a Call object to specify values such as the service ID, method name, and parameters of the method such as attachments. The invoke() operation of the Call object requires the URI of the rpcrouter servlet and returns a Response object. The response is also a SOAP message. If an error occurs during the invocation of the method, a Fault object can be returned within the SOAP message body. The Fault object includes a code and a description of the error.

The SOAPMappingRegistry class handles the marshaling and unmarshaling of data types and comes with preregistered serializers and deserializers such as the MimePartSerializer class. This and other classes and interfaces you might encounter using the Apache SOAP Messages with Attachments implementation are summarized in Table 1.

CLASS/INTERFACE NAME	DESCRIPTION
Call	org.apache.soap.rpc class: Represents remote procedure call
DataSource	javax.activation.DataSource interface: Provides abstraction of some collection of data, a type for that data, and access to it in the form of InputStreams and OutputStreams where appropriate
ByteArrayDataSource	org.apache.soap.util.mime.ByteArrayDataSource class: Implements javax.activation.DataSource and creates DataSource from byte array, File, InputStream, or String
SOAPMappingRegistry	org.apache.soap.encoding.SOAPMappingRegistry class: An XMLJavaMappingRegistry with preregistered serializers and deserializers to support SOAP
BeanSerializer	org.apache.soap.encoding.soapenc.BeanSerializer class: Can be used to serialize and deserialize JavaBeans using SOAP-ENC encoding style
Parameter	org.apache.soap.rpc.Param class: Represents argument to RPC call; parameter objects are used by both client and server
DataHandler	javax.activation.DataHandler class: provides consistent interface to data available in many different sources and formats
MimeBodyPart	javax.mail.internet.MimeBodyPart class: Represents MIME body part
MimePartSerializer	org.apache.soap.encoding.soapenc.MimePartSerializer class: Implements Serializer and Deserializer interfaces and provides functionality to serialize InputStream, MimeBodyPart, DataSource, and DataHandler objects from/to multipart Mime attachments to SOAP message; also provides functionality to deserialize to DataHandler

TABLE 1 Summary of important classes when using Apache SOAP implementation

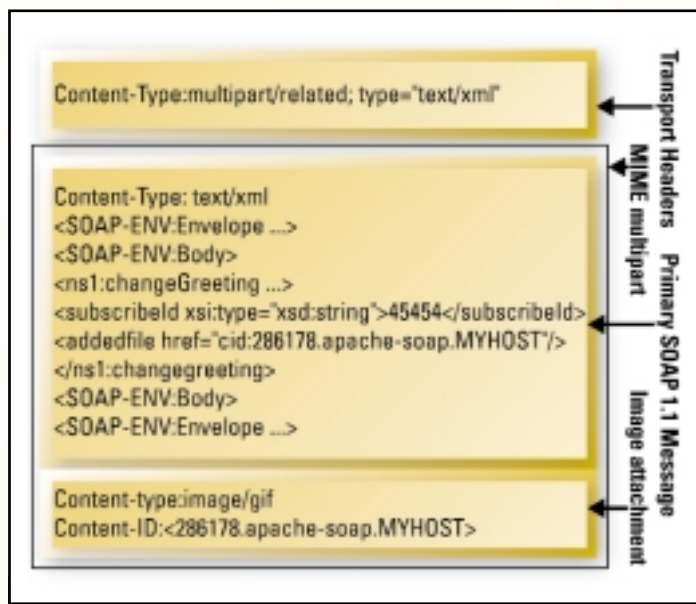


FIGURE 1 Sample structure of a SOAP message package

Example

To provide a simple, concrete illustration of the previous discussion, let's assume we have a subscriber who uses systems provided by messaging system vendors *A* and *B*. Suppose, using *A*'s system, a subscriber needs to change his/her voicemail greeting on *B*'s system. This need can be met by a Web service using SOAP provided by vendor *B*. So, vendor *A* develops a Web application that accesses vendor *B*'s Web service using SOAP to change the voicemail greeting of a subscriber.

To support this scenario, two simple Java programs will be used to illustrate the SOAP RPC approach rather than the message-based approach. The first program will implement the Web service; the second will be a test client to call the service. The class to support the Web service consists of one operation, `changeGreeting()`, with two parameters – a subscriber ID and a voice greeting in the form of a WAV file attachment. The attachment is obtained using a `DataHandler` object and written to disk for this example. Vendor *B*'s existing business objects could then use this audio file to actually change the greeting. Notice how the Web service can simply provide a coarse-grained service-based interface that shields the calling application from an organization's existing business-layer implementation (Java classes, CORBA objects, EJBs) that is reused to support the Web service. A code snippet of the class that implements this Web service is given in Listing 1.

To deploy the code, a deployment descriptor is needed to specify the URN of the SOAP service ("urn:soapattachtest") and operations (`changeGreeting`) that SOAP clients can use. The Java class (`GreetingService`) that supports the Web service is also specified in the deployment descriptor provided in Listing 2.

The other program involved in this example is a client program that calls the Web service. In this SOAP client a `Call` object is created and initialized with the serviceID and the name of the operation (`changeGreeting`), both of which must match the values in the deployment descriptor. The `Call` object is also initialized with two parameters needed for the `changeGreeting()` operation – the subscriber ID and a voice greeting attachment file. After the `Call` object is initialized, the URL representing the `rpcrouter` servlet is passed into its `invoke()` operation to call the Web service and send the attachment. Listing 3 gives a code snippet of this class. The code listings shown can be deployed on Apache Tomcat and SOAP 2.2.

The Apache SOAP implementation also includes a useful utility called `TcpTunnelGui` that allows you to "sniff" SOAP messages exchanged between a SOAP client and a SOAP server. For example, assuming you're using Apache Tomcat, a SOAP client can connect to port 8070 to send SOAP requests that are then routed to port 8080 (Apache Tomcat's default port) on your Web server's host. As you can see

in Figure 2, this tool is helpful for learning about the contents of SOAP requests and responses as they pertain to the SOAP Messages with Attachments specification.

Conclusion

In this article I introduced the W3C note on sending attachments with SOAP messages and its Apache SOAP implementation. This should provide you with an understanding of an underlying Web service technology that enables the exchange of MIME attachments between disparate systems either within or between organizations. While the article presented a simple discussion to apprise you of the SOAP Messages with Attachments specification, I didn't discuss the other components of a Web service architecture for describing, registering, and discovering Web services such as WSDL and Universal Description, Discovery and Integration (UDDI). More information on these emerging standards can be obtained from the **Resources** section.

Using SOAP with attachments enables solutions for integration of disparate systems and provides a viable foundation for emerging XML-based specifications. The JAXM (Java API for XML Messaging) draft specification provides a standard API for applications that use in-

dustry messaging standards based on SOAP. JAXM Profiles are used to provide a foundation for supporting different higher-level industry standard messaging protocols built on SOAP and SOAP Messages with Attachments. See **Resources** for details.

“The invoke() operation of the Call object requires the URI of the rpcrouter servlet and returns a Response object”

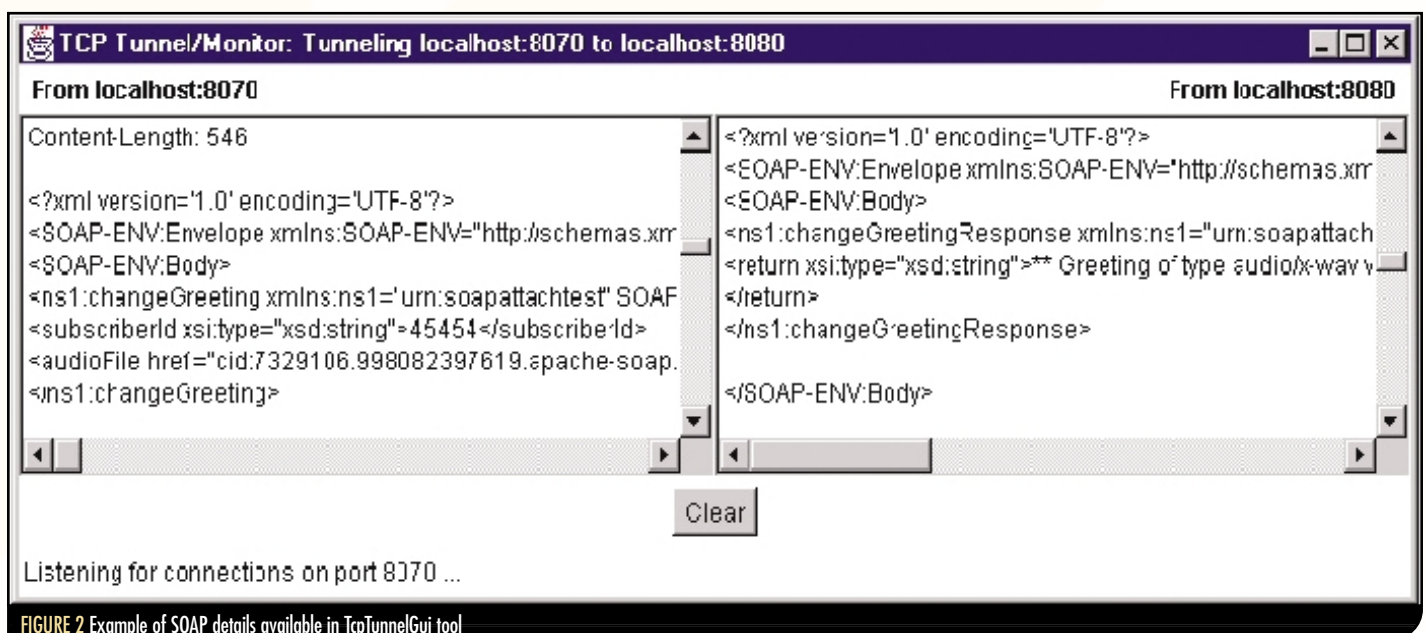


FIGURE 2 Example of SOAP details available in `TcpTunnelGui` tool

Resources

- **SOAP 1.1 Specification:** www.w3.org/TR/SOAP/
- **Apache SOAP:** www.apache.org
- **JavaMail:** <http://java.sun.com/products/javamail/>
- **Java API for XML Messaging (JAXM) Specification:** http://java.sun.com/xml/xml_jaxm.html
- **Web Services Description Language (WSDL):** www.w3.org/TR/wsdl
- **Universal Description, Discovery and Integration (UDDI) project:** www.uddi.org/
- **SOAP Messages with Attachments:** www.w3.org/TR/SOAP-attachments ☒

AUTHOR BIO

Ian Moraes, PhD, leads the development of technical strategies and system architectures of unified messaging solutions for a telecommunications solutions provider. He has architected, designed, and developed systems using J2EE, XML, and WAP.

I MORAES @ YAHOO.COM

LISTING 1 Snippet of Web service implementation class

```
public String changeGreeting(String subId, DataHandler
dh) throws IOException {

    // create response string to send back to caller
    StringBuffer res =
        new StringBuffer("Greeting type : ");
    res.append(dh.getContentType());
    // create a temp file name for the greeting
    StringBuffer tempName = new StringBuffer(128);
    tempName.append("greeting");
    tempName.append(subId);
    tempName.append(System.currentTimeMillis());
    tempName.append(".wav");
    String fileName= tempName.toString();
    // write audio content to file
    try {
        DataSource ds = dh.getDataSource();
        ByteArrayDataSource bds =
```

```
new ByteArrayDataSource(ds.getInputStream(),
dh.getContentType());
FileOutputStream fos = new
FileOutputStream(fileName);
bds.writeTo(fos);
} catch(IOException exc) {
    exc.printStackTrace(System.err);
}
return res.toString();
}
```

LISTING 2 Greeting service deployment descriptor

```
<isd:service
xmlns:isd="http://xml.apache.org/xml-soap/deployment"
id="urn:soapattachtest">
<isd:provider type="java" scope="Request"
methods="changeGreeting">
<isd:java class="xmlj.GreetingService"
static="false"/>
</isd:provider>
</isd:service>
```

LISTING 3 Snippet of Web service client class

```
URL url = new URL(routerUri);
SOAPMappingRegistry smr = new SOAPMappingRegistry();
// Initialize call object
Call call = new Call();
call.setSOAPMappingRegistry(smr);
call.setTargetObjectURI(serviceUri);
call.setMethodName(methodName);
call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
// Initialize method parameters
Vector params = new Vector();
params.addElement(new Parameter("subscriberId",
String.class,subscriberId, null));
DataSource ds = new ByteArrayDataSource(new
File(audioFile), null);
DataHandler dh = new DataHandler(ds);
params.addElement(new Parameter("audioFile",
javax.activation.DataHandler.class, dh, null));
call.setParams(params);
// Invoke the operation.
Response resp;
resp = call.invoke(url, "");
```

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

Special online offer

PICK 4

Subscribe

for One Special Low Price

WWW.SYS-CON.COM



Wireless Business & Technology

Java Developer's Journal

Web Services Journal

XML-Journal

WebLogic Developer's Journal

WebSphere Developer's Journal

ColdFusion Developer's Journal

PowerBuilder Developer's Journal



OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SYS-CON
MEDIA

Begin your unwired experience now



SYS-CON Media, the world's leading publisher of Internet technology magazines for developers, software architects and e-commerce professionals, becomes the first to serve the rapidly growing wireless application development community!

information for the unwired world

www.wbtz.com

*Encrypting XML documents dynamically*

Using the IBM XML Security Suite

Part 2 of 2

Someday soon you'll be able to tell the Internet what you're interested in and have it respond with the information you need. You could instruct a Web site to monitor traffic reports, for example, then tell it where you're heading. If there's a traffic jam, the Web site will dial your cell phone to alert you and suggest an alternate route.

Last month I revealed a way to implement security for your XML documents. The advantages of adding security to XML documents were discussed as well as the World Wide Web Consortium working drafts that cover XML encryption.

We explored how to use the IBM XML Security Suite toolkit to encrypt XML documents – how to do a simple encryption of an XML document and then decrypt the same document. The toolkit uses standards that are in working draft form from the W3C.

To perform XML signatures, I'll use the same tools I used before, but to make things interesting I'm going to switch to the Xerces Java XML parsers to show you the flexibility of the toolkit. (You'll recall that in Part 1 I used Sun's XML parser, JAXP.) I'm still going to use the IBM Java Cryptography Extension, for its RSA implementation, and Xalan, which can be found at <http://xml.apache.org>. Of course, I'm using the same XML document, insurance.xml. See Listing 1 for complete XML document.

The keystore that was created in Part 1 will be used here too. Since we created a key pair using RSA, we'll use the same keys to digitally sign our XML document.

Digital Signatures

Digital signatures offer authentication and integrity. With a signature comes assurance that a message hasn't been tampered with and that it originated from a specific person. One thing to remember is that signatures don't provide confidentiality. The signature accompanies a plaintext message that anyone can intercept and read. Using the signer's private key generates a signature and only the signer's public key can decrypt the signature, which is the authentication. For

integrity to be assured, the message digest of the message must match the decrypted message digest.

Digital signatures require the use of Public Key Infrastructures, PKI. Without going into much detail, with PKI there are two keys, a public one and a private one. The user publishes the public key and the data is encrypted using his or her private key to create what is known as a *hash*. Decryption of the same data uses the user's published public key.

Digitally signing an XML document is a difficult task. One neat thing about digital signatures is that an application may sign parts of the document but still allow updating.

W3C Working Drafts

The current status of the W3C XML Signature Syntax is in its second candidate recommendation and will be promulgated as a Proposed Recommendation and Draft Standard. I can't stress enough that you review this syntax and processing document. It provides a detailed explanation of the elements required to accomplish XML signatures. It also can help you understand why some of the APIs work the way they do and the order in which to configure the toolkit.

Configuring the Signature DOM

First we start by preparing a Document Object Model using the SignatureGenerator class. There are three algorithm parameters for the SignatureGenerator instance: the default digest algorithm, the canonicalization algorithm, and the signature algorithm. The other arguments are the document factory and a KeyInfoGenerator, which we're passing as null.

The default digest algorithm is the URI used for the DigestMethod elements

in the signature. This is the element that identifies the digest algorithm to be applied to the signed object. If you don't specify an algorithm URI to a Reference instance, this algorithm is used.

The next important parameter is the canonicalization parameter. This algorithm URI is for the canonicalization of the SignedInfo element, which includes the canonicalization algorithm, a signature algorithm, and one or more references. This element may contain an optional ID attribute that will allow it to be referenced by other signatures. Remember this, for this ID will come into use later.

The final algorithm used in the SignatureGenerator is one used for signing the SignedInfo element. Your SignatureGenerator code would look like this (see Listing 2 for detail breakdown):

```
SignatureGenerator signatureGen = new  
SignatureGenerator (docFactory,  
DigestMethod.SHA1, Canonicalizer.W3C,  
SignatureMethod.RSA, null);
```

The next step in setting up the DOM will be to register the Reference instances. To keep things short and sweet, we'll use a Reference object for an enveloping signature.

To begin with, you'll have to create an Object element and you can use the wrapWithObject() method to do this. wrapWithObject() takes two arguments, a Node and an ID. The Node in this case is the Node that's going to be embedded in the signature. The ID is a URI that will identify the data object to be signed. This is an optional attribute of the Reference element, but we'll use it.

```
Element objectElement =  
signatureGen.wrapWithObject  
(xmlDoc.getDocumentElement,  
"RefID");  
Reference ref = signatureGen.  
createReference(objectElement);
```

AUTHOR BIO

Joseph Smith has been designing and developing Web-based solutions for the last five years.

JavaOne

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

SUBSCRIBE AND SAVE

XML JOURNAL

Offer subject to change without notice

ANNUAL NEWSSTAND RATE

~~\$82.88~~

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



Following the establishment of the Reference element, the Transforms element must be created. This element is an ordered list that describes how the signer obtained the data object that was digested. Each Transforms element contains an Algorithm attribute; we'll use the W3C Canonical XML by XML Signature Working Group, W3CC14N2. To set transformations to a Reference instance, use

```
Reference ref = sigGen.createReference(
    (objectElement));
ref.addTransform(Transform. W3CC14N2);
```

The final step in preparing the DOM tree is to call getSignatureElement(). This will return the Signature element, but this element isn't a child of any element or document.

```
Element signatureElement = sigGen.
    getSignatureElement();
```

Signing the Document

The SignatureContext object will perform the signing of the document. Here's where we use the sign() method of the SignatureContext object to perform the digital signature.

Remember, we're using the keystore that we set up in Part 1. Using that keystore, retrieve the X509Certificate using the KeyInfo object and pass the certificate to an instantiated X509Data object.

```
X509Data x5data = new X509Data();
x5data.setCertificate(cert);
x5data.setParameters(cert, true,
```

```
true, true);
keyInfo.setX509Data(new
    KeyInfo.X509Data[] { x5data });
```

Finally, prepare a key used to sign and call SignatureContext.sign(Element, Key) – we're using the key from the keystore. Keep in mind that keystore contains the RSA key pair.

```
new SignatureContext()
    .sign(signatureElement, key);
```

Verifying a Signature Document

The toolkit provides a GUI for testing whether the digital signature worked. VerifyGUI reports the validity of each resource and of the signature. If the signature and all of the signed resources weren't modified, VerifyGUI reports the result of verification as "Core Validity: Ok".

Conclusion

As I mentioned in Part 1, this is all alpha. Since things could change with W3C, the toolkit has to be flexible for those changes as well.

Currently, two JSRs supply Java APIs that accomplish tasks similar to those accomplished by the IBM XML Security Suite.

References

- JSR 105 XML Digital Signature API: <http://jcp.org/jsr/detail/105.jsp>
- W3C/IETF XML Signature specification: www.w3.org/2000/09/xmldsig

JSMITH @ PATRIOT.NET

LISTING 1 insurance.xml

```
<?xml version="1.0" ?>
<insurance>
  <health carrier="Armor">
    <memberinfo>
      <name>Joseph Smith</name>
      <subnumber>bmc2000</subnumber>
      <group>00001</group>
      <copay>
        <office>15.00</office>
        <medicine>
          <brand>10.00</brand>
          <generic>5.00</generic>
        </medicine>
      </copay>
      <dependents>
        <depname>wife</depname>
        <depname>child</depname>
      </dependents>
    </memberinfo>
  </health>
  <dental carrier="none" />
  <vision carrier="none" />
</insurance>
```

LISTING 2 XMLSign.java

```
import java.io.*;

import java.security.*;
import java.security.cert.*;

import org.apache.xerces.dom.DocumentImpl;
```

```
import org.apache.xerces.parsers.
    DOMParser;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

import com.ibm.xml.dsig.*;
import com.ibm.xml.dsig.KeyInfo.*;
import com.ibm.dom.util.ToXMLVisitor;
// use for output

/**
 * XML signer
 *
 */
public class XMLSign {
    private Key key;
    private String xmlFile =
        "insurance.xml";
    private String keyName =
        "xml.keystore";
    private String keyPass =
        "xmlpass";
    private String keyAlias =
        "mykey";
    //private SignatureGenerator
    signatureGen;
    Element signatureElement;

    static void setupReference
    External(SignatureGenerator
    signature, String uri)
    throws IOException {
```

```

Reference ref = signature
.createReference(uri);
signature.addReference(ref);
}
public void doSign()
throws Exception
{
// 1. Retrieve keys from
keystore
KeyInfo keyinfo =
getKeyStore();
// 2. Configure Security
Suite
configure(keyinfo);
// 3. Sign and write the
XML document
signIt();
}

private KeyInfo getKeyStore()
throws FileNotFoundException
Exception, IOException,
CertificateException,
KeyStoreException,
NoSuchAlgorithmException,
UnrecoverableKeyException,
SignatureStructureException
{
KeyStore store = KeyStore
.getInstance("JKS");
store.load(new FileInputStream(
keyName), keyPass
.toCharArray());
key = store.getKey(
keyAlias, keyPass.to
CharArray());
// a private key

X509Certificate cert =
(X509Certificate)store
.getCertificate
(keyAlias);
KeyInfo keyInfo =
new KeyInfo();
X509Data x5data = new
X509Data();
x5data.setCertificate
(cert);
x5data.setParameters
(cert, true, true, true);
keyInfo.setX509Data(new
KeyInfo.X509Data[]
{ x5data });
keyInfo.setKeyValue
(cert.getPublicKey());
return keyInfo;
}

private void configure
(KeyInfo keyinfo)
throws Exception
{
Document doc =
new DocumentImpl();

SignatureGenerator
signatureGen = new
SignatureGenerator
(doc, DigestMethod.SHA1,
Canonicalizer.W3C, SignatureMethod.
RSA, null);

signatureGen.setKey
InfoGenerator(keyinfo);
configureReference
(signatureGen);
}

private void signIt()
throws Signature
StructureException,
NoSuchAlgorithmException
Exception, NoSuchProvider
Exception, InvalidKey
Exception,
SignatureException,
TransformException,
IOException
{

```

```

new SignatureContext()
.sign(signatureElement,
key);
writeElement
(signatureElement);
}

void configureReference
(SignatureGenerator sigGen)
throws IOException{
// parse xml file
// get the parser and
parse the file
try
{
DOMParser parser =
new DOMParser();
parser.parse
(xmlFile );
Document xmlDoc =
parser.getDocument();
Element objectElement
= sigGen.wrapWith
Object(xmlDoc.get
DocumentElement(),
"RefID");//id;

Reference ref =
sigGen.create
Reference
(objectElement);
ref.addTransform
(Transform.W3CC14N);
sigGen.addReference
(ref);
signatureElement =
sigGen.getSignature
Element();
}
catch (SAXException sax)
{
saxe.printStackTrace();
}
}

private void writeElement
(Element el)
{
try {
Writer wr = new
OutputStreamWriter
(System.out, "UTF-8");
wr.write("<?xml
version='1.0'
encoding='UTF-8'>\n");
new ToXMLVisitor(wr)
.traverse(el);
wr.close();
} catch (Exception ex) {
ex.printStackTrace();
}
}

public static void main
(String[] argv)
throws IOException,
KeyStoreException,
CertificateException,
NoSuchAlgorithmException,
InvalidKeyException,
SignatureException,
UnrecoverableKeyException,
TransformException,
SignatureStructureException,
NoSuchProviderException{

try
{
XMLSign sign = new
XMLSign();
sign.doSign();
} catch (Exception e) {
e.printStackTrace();
System.exit(1);
}
}
}

```

www.sys-con.com/xml

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 January JDJ:

J2ME Benchmarking: A Review

Evaluating an application's performance objectively

J2EE Application Security

Container- versus application-managed security

J2EE Testing: What's It to You?

An overview of the CTS for J2EE component developers

Got a Problem with J2EE?

Doctor Java has a cure

The Key to Superior EJB Design

The specifics of EJB and how they influence the design of EJB systems





Here's help when your applications run into bottlenecks

XML Enabled Applications: Need for Speed

XML has found numerous applications as a format for representing content and modeling business objects, as a messaging format, and as a data source. In applications that use XML in this capacity performance often leaves room for improvement. Conventional methods of processing XML prove to be inadequate and it becomes necessary to explore other options to improve the speed of your application.

This article identifies some common operations that are performance hogs and finds ways to optimize these operations, thereby improving the overall performance of an application.

Identifying Bottlenecks

Let's consider a few common application scenarios and attempt to isolate those generic XML operations that show room for optimization. In the discussion of each scenario I'll identify the XML technologies and standards involved, find the bottlenecks in XML processing, and then explore ideas for easing them.

Technologies and standards related to XML have found acceptance in a wide variety of applications. Some of the more popular of these include:

- **Web services:** Initiatives like UDDI, with supporting specifications like WSDL, SOAP, and so on, provide a framework

for registering, discovering, and interacting with Web services deployed by businesses.

- **Content management:** XML can be used as a format for storing content. This allows us to impose structure on the content (DTDs, XML Schema), make the content easily searchable (XPath, XQuery), improve the quality of the searches (RDF), and provide multiple content delivery options (XSL).
- **B2B applications:** XML vocabularies have been defined to ease interoperability of otherwise disparate B2B applications (e.g., RosettaNet, ebXML, CBL).
- **Wireless portals:** Service providers need their applications to be device agnostic and XML fits the bill as a device-, platform-, and programming language-agnostic message format. XML essentially gives us a write once, render everywhere solution that is

important for a wireless application whose clients are a variety of mobile devices such as phones and PDAs.

A couple of sample scenarios of these applications will help illustrate the performance problems and their possible solutions.

Scenario 1 – Content Providers

Content providers can use XML effectively as a format for storing and distributing content. The advantages a content provider can derive from using XML include the ability to:

1. Impose a schema on the content, thereby bringing structure to the content (DTDs and XML Schema documents).
2. Have one source document format and multiple content delivery formats (XSL – content can be delivered as HTML for browsers, WML for mobile devices, MS Word or PDF documents, or XML for other client applications, etc.).
3. Personalize content, have better search capabilities by including relevant metadata with the document

AUTHOR BIO

Srinivas Pandrangi is a technical architect at Ipedo, Inc., a company involved in developing high-performance dynamic content delivery products to accelerate Internet and wireless applications. Srinivas has participated in standards definition and their implementation related to XML, LDAP, X.500, and PKIX through the W3C and IETF.

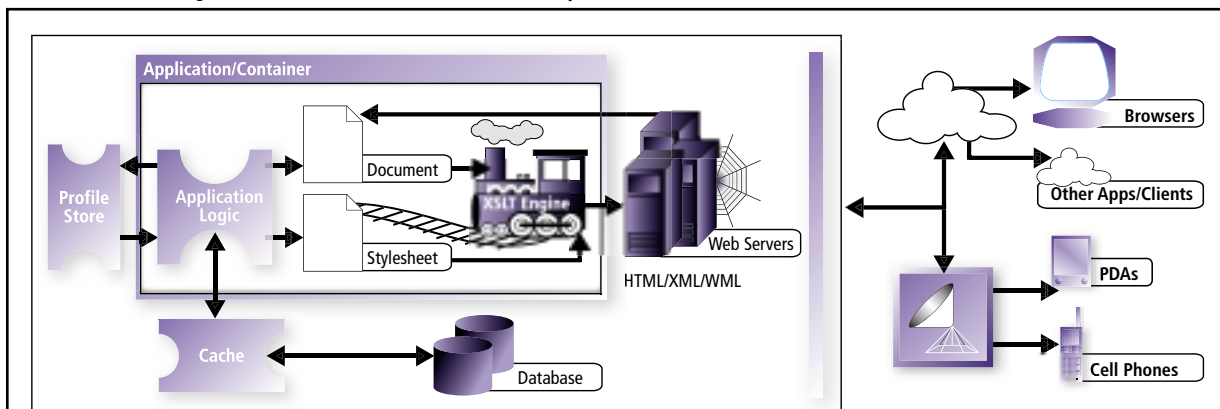


FIGURE 1 Profile-driven content delivery

(RDF) and easier updating of the content, since it's more structured.

In the current state of XML's evolution, it's not yet common for clients to have XSL processing capabilities. This places a requirement on the server side to perform the transformations. Any Web site receiving a few hundred thousand hits an hour will face an extremely difficult problem with respect to response times. An excellent example of this scenario is IBM's Web site at www-106.ibm.com/developer/works/library/xml-handbook/xml-at-ibmcom/index.html.

Figure 1 illustrates a simplified content provider scenario. The content is stored as XML documents in a datastore, as are stylesheets that are used to render this content in accordance with a client's preferences. When the application receives requests for content, it retrieves the XML document containing the requested content. Based on the profile of the client/user requesting the content, it also retrieves an appropriate stylesheet. It transforms the content XML document using the stylesheet and delivers the results to the client. The application attempts to cache the documents retrieved by previous requests in an effort to improve performance.

In the absence of any knowledge of the actual application logic, we can examine the application for bottlenecks from a generic XML processing viewpoint.

Parsing

Parsing a document involves traversing the content of the document and creating an in-memory representation (DOM) or processing events corresponding to the structure of the document as they are encountered (SAX). In addition, if the document is to be validated, the structure of the document has to be reconciled against a schema represented by a DTD or a schema document. This requires parsing the DTD or schema document as well. All in all, parsing is an expensive operation that should be done as infrequently as possible. This conclusion implies that the parsed state of a document should be preserved for a subsequent request. The application depicted in Figure 1 attempts to do this by caching the parsed document. This approach is suitable when dealing with a limited number of documents, but the demands on memory resources can be quite significant for most practical deployments.

A more suitable approach to preserving the parsed state of the document is PDOM (Persistent DOM). In a PDOM implementation the parsed state of the document is persisted. If the application

developer wants to have access to the parsed document without placing the document in memory, this would be an ideal solution. It is similar to the Deferred DOM implementation of the Xerces parser, except that it persists the DOM to disk, so there is no need to parse the document for any future request.

Transforming

An XSLT processor accepts as input a source XML document and an XSLT stylesheet and performs a transformation to create a target document. An XSLT stylesheet contains instructions on how the source document is to be transformed into the target document. Transformations have been found to be the source of agonizing performance problems. Two approaches can be taken to ease this problem.

- Reduce the number of transformations to the minimum number possible.
- Optimize the transformation process.

The first approach implies either caching the results of the transformation or pretransforming the content (as proposed in the case of the [ibm.com](http://www.ibm.com) Web site case). This, again, is a solution that works well for a small number of documents or a small number of transforms, but it won't scale well for situations involving a large number of documents, a large number of transforms, and a large number of output documents. For example, if personalization is required, pretransforming the content is ruled out since we can't create a version of the document for every user of the system.

A number of different approaches can be taken to optimize the transformation process itself. By using a scheme like PDOM, we can optimize the first step, loading the source document and stylesheet. Tackling the next step, we can generate and preserve the execution plan for the transform. The plan can be preserved as a Java class or in some other application-specific form. Xalan provides support for Translets (XSLTC – see http://xml.apache.org/xalan-j/xsltc_usage.html), a bunch of Java classes, which are compiled representations of the XSLT stylesheet. Sun is also purportedly working on its own XSLT compiler.

One piece that can be optimized still further is the XPath Query processing. XSLT stylesheets use XPath expressions to address parts of the source document. (Ways to optimize XPath query processing are addressed later in this article.)

Scenario 2 – B2B Applications

Common uses of XML in B2B applications are as a message format and as

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:

i-mode 101

A WBT Special for all wireless operators and developers around the globe hoping to quickly learn how to imitate NTT DoCoMo's wildly successful i-mode data offering in Japan.

Is Wireless Broadband Barreling Your Way?

WBT's David Geer looks at wireless broadband connectivity as it spreads around the globe.

The Impact of Privacy on the Future of Wireless Advertising

WBT's SMS editor, Dan Lubar, questions the balance of location-based advertising and an individual's right to privacy.

The GSM Association M-Services Initiative

A look at a series of guidelines generated to give WAP a friendlier consumer face.



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 January WSJ:

Invoking .NET Web Services from Mobile Devices
Pocket PCs put .NET Web services at your fingertips

It's More Than Just the Plumbing
The real issues are the nontechnical ones

Wireless Web Services with J2ME
Remote possibilities

Web Services – Tiptoeing Through the Snarl
Creating Web services with unprecedented uptimes

Bringing Web Services to Smart Devices
Realizing the value of the connected world

Web Services @ Work
Gluing Web services to Baan

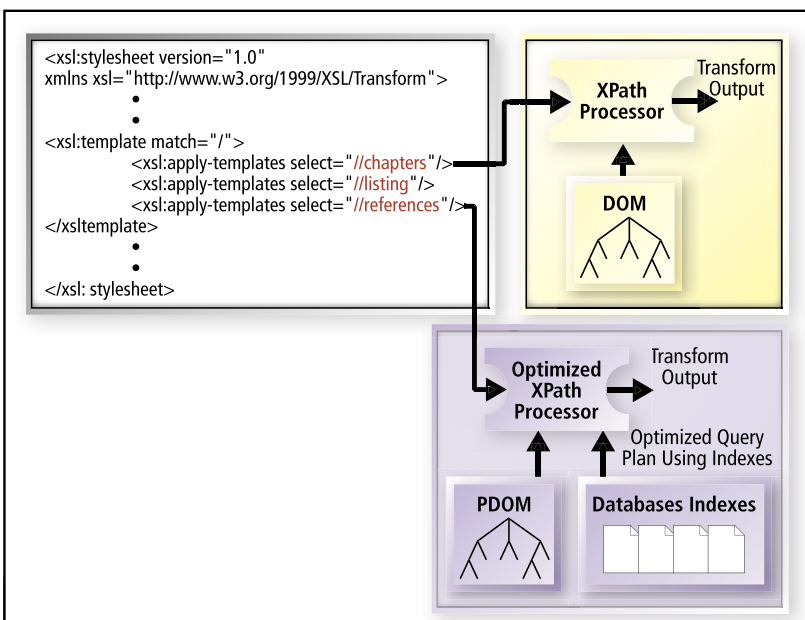


FIGURE 2 Optimized transforms using indexes

a format for business documents. These applications need to perform computationally expensive operations such as transforming, querying, and archiving XML documents. The previous discussion of parsing and transforming bottlenecks is also pertinent to this scenario. In addition, B2B applications also need to query document bases on a much more extensive scale than a content provider would. For this reason a B2B application might consider storing the XML documents in a relational database. This leads us to explore two other XML database operations that can use optimization.

Querying

A number of operations in B2B applications result in queries being performed against the document bases. As mentioned before, XSLT transformations depend on evaluation of XPath expressions. Most XPath engines assume an underlying DOM implementation, and every expression evaluation involves a brute-force approach that involves traversing the document. Such an approach can be avoided if indexes are available that can optimize the query processing. The ability to build indexes requires that the XML documents be stored within the purview of a database management system (see Figure 2).

Storing Data

When archived, XML documents are decomposed into a format that is useful for storing in the underlying database, unless the underlying database is a native XML database. For example, storing an XML document in a relational

database requires a conversion from a hierarchical model of data to a relational model and vice versa for retrieval. To avoid conversions between the hierarchical XML documents and the nonhierarchical database format, we need a database that's capable of storing the XML documents in their native format. The database should also be capable of processing queries expressed using standard XML query languages like XPath and XQuery.

However, given that most enterprises have an entrenched database management system, a more practical solution is to implement a caching layer on top of the database, which gives an "XML view" of the data. The cache needs to be synchronized with the database, and needs to perform some other functions that a database performs, like (XPath) query processing and transaction management. XML databases available today either have this functionality already, or are moving toward supporting such functionality.

Conclusion

XML-enabled applications need to perform computationally expensive operations on a regular basis, just given the nature of XML. To exploit the usefulness of XML it's necessary first to ensure acceptable performance of these applications. This might require that the application look beyond the conventional ways of processing XML to find more efficient options, such as using PDOM, compiled transforms, and XML databases. ☐

SPANDRAN @ IPEDO.COM

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



DataJunction Integration Suite

by DataJunction

— [REVIEWED BY H. RUSSEL DOUGLAS] —



Bio

H. Russel Douglas has been CTO/CIO of both established companies and start-ups. He is currently a principal with NECG, where he and a team of fellow CXOs offer temporary CIO/CTO/COO services to small and mid-size companies.

RUSSEL.D @ SWBELL.NET

Do you remember those electronic sets where you took various colored bits, followed a set of simple instructions and voilà, you had a radio. Well, at least some people got radios. I always seemed to end up with a few bits left over. So imagine my surprise when I plugged my bits of data together with Data Junction and it worked.

Data Junction, also the company name, is a set of tools to translate and integrate data. It seems that no matter which customer site I've visited recently, one of the departments is wrestling with mapping or converting data. The suite's components are Data Junction, Integration Engine, and Content Extractor.

This review focuses on the very real world problem of data extraction and conversion. The Data Junction tools are designed to extract data from one source, massage it into a new form, and insert it into a second source. Such programs succeed or fail in the richness of their data-mapping tools. In this area Data Junction seems to cover most of the bases. I was able to identify over 150 data and application formats, including XML, that are provided for in the product for bidirectional manipulation and transformation.

The process to complete a conversion is made simple by the visual interface that is part of the product offering. Users consider the process as three easy and distinct steps. The user interface presents the steps as three tabs. Under the Source tab source data is identified and a format chosen. During this process the user fills in administrative detail, such as database or file location, user name, and password. From the Target tab the user then sets up the desired final data structure. Again, administrative details may be required. The final step is to create a set of rules to perform the transformation; this is done from the Map tab.

The UI is intuitive and I was able to create a number of simple conversions in a matter of minutes. In fact, many simple conversions can be accomplished with simple drag-and-drop actions. The real power, however, lies in the ability to create structured data tables from multiple sources, including unformatted text files.

Behind the scenes the extraction process for unstructured data is performed by the Content Extractor. The extraction of desired fields from the source text file is accomplished by visually marking up the file in the Content Extractor user interface. Dialogs appear on the screen allowing the user to express a rich set of pattern recognition rules and actions to assist in the extraction of clean data.

Several techniques are available to view samples of extracted data before converting to an export file. Apart from scrolling the full text of the data, a debug window can be used to

search for all lines satisfying certain extraction criteria. In addition, users can open the source browser to examine the fields and records – which are shown in a row and column format – to see how the data will export.

Included with Content Extractor are the facilities to deal with the following as source data:


- HTML, XML, and other structured documents from the WWW
- E-mail header and body
- Printouts from programs captured as disk files
- Reports of any size or dimension
- ASCII or EBCDIC text files
- Spooled print files
- OCR output
- Complex multiline files
- Downloaded text files (e.g., news retrieval, financial, real estate)
- Online data feeds
- EDI/X12
- Files with tagged data fields

The software installed cleanly, although the installation dialog and interface are less than intuitive. I was unclear what options to install to accomplish a particular goal. After resorting to reading the online manual, I was able to accomplish the task.

I set about creating a Web page that combined a news feed, OCR from a fax, and a set of tables from a SQL database. The process was easy and intuitive. While this example is somewhat contrived, it demonstrates the breadth of input sources that can be combined.

Users needing to tackle real-time transactions leverage Data Junction Enterprise Edition for significantly improved performance. Enterprise Edition offers a multithreaded Integration Engine that runs multiple complex conversions in unison, thus speeding transaction time immensely.

While the documentation is adequate for seasoned data professionals, Data Junction could be useful to a much larger audience than is targeted by the documentation in its current form.

Overall, I'd rate Data Junction's product as superior. It's powerful and easy to use, and customers seeking a simple tool to accomplish complex data transformations would do well to investigate it. Performance could be an issue for users who need to perform complex conversions in real time, but this is only a small portion of the potential user base. 

SPECS

Data Junction Corporation

2201 Northland Drive
Austin, TX 78756-1117

Phone: 800 580-4411

Web: www.datajunction.com



Test Environment

OS: Windows 2000 Advanced server with SP2
Processor: Pentium III 450MHz
Memory: 512MB

Specifications

Platforms: WIN, NT, Linux, Sun Solaris,
IBM-AIX, HP-UX

Pricing: Less than \$850 for single-user personal license; pricing increases with platform power for enterprise licenses

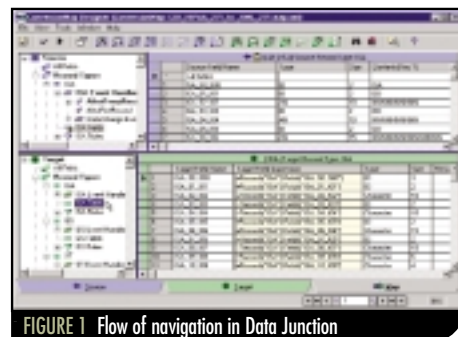


FIGURE 1 Flow of navigation in Data Junction

XML-J ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com		68
BEA eWorld	www.bea.com/events/eworld/2002	404-240-5506	19
DataMirror	www.datamirror.com/resourcecenter	800-362-5955	6
HitSoftware	www.hitsw.com.com	800-345-4001	9
IONA	www.iona.com/ws-webcasts		2
Java Developer's Journal	www.sys-con.com	800-513-7111	55
JavaOne	http://java.sun.com/javaone	888-886-8769	53
JDJ Store	www.jdjstore.com	888-303-JAVA	27, 34, 35
Macromedia	www.macromedia.com/downloads	888-939-2545	13
Software AG	www.softwareagusa.com/XML4U	877-SAG-4XML	67
Sonic Software	www.sonicsoftware.com		3
SYS-CON Custom Media	www.sys-con.com	925-244-9109	44
SYS-CON Media	www.sys-con.com	800-513-7111	50, 61
SYS-CON Reprints	www.sys-con.com	201-802-3026	31
Web Services Conference & Expo	www.sys-con.com	201-802-3069	32, 33, 39
Web Services Journal	www.sys-con.com	800-513-7111	58
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	800-513-7111	45
WebSphere Developer's Journal	www.sys-con.com/websphere	800-513-7111	29
Wireless Business & Technology	www.sys-con.com/wireless	800-513-7111	51, 57
Wireless Edge Conference & Expo	www.sys-con.com	201-802-3069	41
XML Global	www.xmlglobal.com/newangle	800-201-1848	4
XML-Journal Technologies	www.sys-con.com	800-513-7111	54

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.webspheredevelopersjournal.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

Applied Templates

Simplify XSLT code with applied templates

by Roy Houbler

XML Development Environments

How to create a source environment for XML

by Jim Gabriel

Q&A: DIVE into XML

by Trace Galloway

I Am the Official Mascot of XSLT

A transforming robot looking for work

by Tod Emko

Intelligent XML Content Firewalls

Semantic-based filtering of digital content

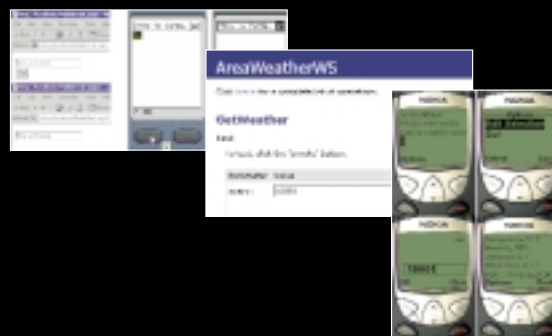
by John R. Callahan

Got XSLT? - PART 4

How to transform XML to SVG easily with style

by Shouki Souri

Next Month...



XML JOURNAL

Don't miss the February issue!

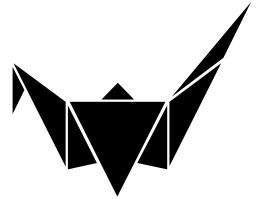
XML Training

Crane Softwrights

Lecture and hands-on training materials in XML-related Recommendations can be licensed for both public and in-house use. See <http://www.CraneSoftwrights.com/xj/schedall.htm> for syllabi and deliveries by G. Ken Holman, <http://www.CraneSoftwrights.com/xj/sched.htm> for deliveries in Ottawa (including "train the trainer" for licensees), and

<http://www.CraneSoftwrights.com/xj/licensing.htm> for licensees who can deliver Crane's courses at your site.

www.CraneSoftwrights.com/xj
(613) 489-0999



DMSi - Document Management Solutions, Inc.

DMSi presents a full suite of solutions-based training services focused on the specific needs of your environment. Courses include data architecture development, XML, XSL/T and related standards, and product-specific training. We customize these courses to include your data and scenarios related to your organization. The result is a class that

satisfies the specific training requirements of your staff.

www.dmsi-world.com/training.htm
(978) 738-9770



ISOGEN International

Acquire practical XML skills the way the professionals do, with either full track or individual classes based on the ISOGEN XML Learning Framework™. Our pragmatic and highly refined approach, our world-class instructors, and our excellent materials guide you to a higher level of XML skill. Open enrollment, onsite training, and customized courses available.

www.isogen.com or www.datachannel.com
(612) 961-6203



OMNIMARK Technologies

OmniMark Technologies, in conjunction with Stilo Technology in the UK, offers a broad range of XML-related courses covering both design and processing. Courses are offered at OmniMark facilities in Boston, Paris, and Leuven, and at Stilo's facilities in London. Custom and on-site training are also available. For more information see our Web site.

www.omnimark.com or www.stilo.com
(613) 745-4242



Online-Learning.com

Offers online courses for professional development; learn XML, XSL, XSLT, Technical Writing, and Usability. All courses provide extensive instructor support, interaction, practical hands on experience, and certification.

Let us teach you how to deliver single source content.

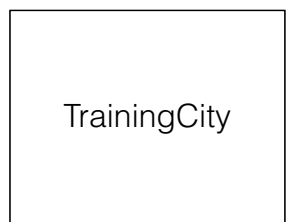
www.online-learning.com/xmljournal.html or
info@online-learning.com (613) 596-5673



TrainingCity

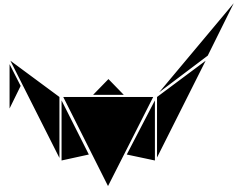
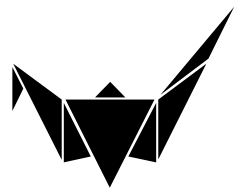
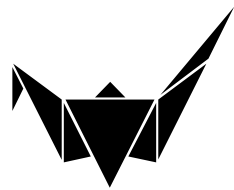
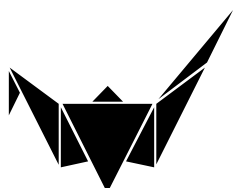
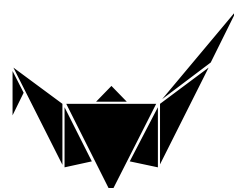
TrainingCity provides XML Consulting services & hands on XML/XSLT training, workshops & elearning. Customized onsite training, CDROMs, & Videos. Call to speak with one of our XML Experts: 858-755-8868, <<mailto:info@TrainingCity.com>> or www.TrainingCity.com for course outlines & free online sample training!

www.TrainingCity.com
(858) 755-8868



For listing opportunities contact
Megan Ring (201) 802-3023 megan@sys-con.com

January 2002

Monday	Tuesday	Wednesday	Thursday	Friday
	1	2	3	4
7	Crane Softwrights: Introduction to XSLT and XPath Internet-live (world-wide access) 	9	Crane Softwrights: Introduction to XSLFO Internet-live (world-wide access) Crane Softwrights: Introduction to XSLT and XPath Live audio instruction to a world audience over the Internet 	Crane Softwrights: Introduction to XSLFO Live audio instruction to a world audience over the Internet 
Online-Learning.com: Professional Technical Writing, Intro to XML, Professional XML Authoring, XSL Techniques, Usability Testing, Information Design Forum Classes available online ONLINE-LEARNING	15	Crane Softwrights: XML in the Corporation Live audio instruction to a world audience over the Internet 	17	Crane Softwrights: XML Stylesheets and Transformations Live audio instruction to a world audience over the Internet 
21	22	23	24	25
28	29	30	31	

XML NEWS

XML-J's Sister Magazine Ranked First in Survey

(Montvale, NJ) – Evans Data Corporation has ranked *Java Developer's Journal* as “the most trusted developer publication” among developers who use Java, according to SYS-CON Media, the world's leading i-technology publisher.



The research results were published in the Evans Data Developer Marketing Patterns 2001

Annual Report, an independent market research report prepared by the leading market research firm serving software and developer markets.

“We are very pleased to see, in our sixth year of publication, that *Java Developer's Journal* continues to serve the fast-growing Java developer community as the hands-down leader of quality Java information in the world. *JDJ's* unmatched leadership is not based solely on its circulation – which is larger than all other Java publications put together – but also on the high quality of its editorial content,” said Alan Williamson, editor-in-chief of the magazine. “The Evans Data report confirms what we hear from our readers and *JDJ's* sponsors and advertising partners every day.”



www.sys-con.com

iWay Upgrades Enterprise Integration Suite

(Toronto) – An enhanced product line designed to lower the implementation and maintenance cost of enterprise integration and improve integration-related project delivery schedules by at least 50% has been announced by iWay Software.

iWay v5.1 is a family of integration software that allows businesses to build a comprehensive and durable infrastructure using

prefabricated middleware products. Its primary products – XML Transformation Server (XTS), Enterprise Integration Broker (EIB), and ETL Manager – are integration servers that centralize and coordinate real-time, near real-time, and scheduled activities between collaborating information resources.



www.iWaysoftware.com

HNC Software to Release Blaze Advisor 4.0

(New Orleans) – HNC Software Inc. has announced the upcoming release of Blaze Advisor version 4.0, which features enhancements designed to provide advanced control over decision-making policies, practices, and procedures in automated systems.



Major technical improvements include decision table support and production debugging.

www.hnc.com

XYZFind Corp. Updates XML Database

(Seattle) – XYZFind Server version 2.0 from XYZFind Corp includes improvements to the XML indexing and retrieval functionality and enhanced performance, scalability, and query capabilities.



www.xyzfind.com

Arbortext, Interwoven Offer Integrated Solution

(Ann Arbor, MI) – Arbortext, Inc., has announced an integration between Interwoven's TeamXML software and Arbortext's Epic Editor. This integrated solution provides customers with the ability to create customized documents that leverage content created in many formats for automatic publishing to the Web, PDF, CD-ROM, and wireless devices.



www.arbortext.com

Flashline Software Enables, Promotes Reuse Initiatives

(Cleveland) – Flashline is now shipping Flashline CMEE version 3, a comprehensive solution that enables, promotes, and measures reuse, offering companies the potential to save millions of dollars in IT expense. To increase a company's return on investment in its software reuse initiative, CMEE now supports and man-

ages all types of corporate software



assets, including Web services, components, and frameworks.

www.flashline.com

XyEnterprise Integrates with Interwoven TeamXML

(Reading, MA) – XyEnterprise has integrated XML Professional Publisher with Interwoven's flagship product, TeamSite, and TeamXML.

The integration allows users to publish XML content to PostScript and PDF while simultaneously publishing the same XML content to the Web. Under the terms of the agreement, XyEnterprise and Interwoven will work collaboratively to bring the integration to the marketplace.



www.interwoven.com
www.xyenterprise.com

New Chutney PreLoader Version Available

(Atlanta) – Chutney Technologies has announced the availability of Chutney PreLoader version 4.0.



Key features of the new version include enhanced security, backup/recovery, and XML and failover capabilities.

www.chutneytech.com

BLOX Available from Infoteria

(Boston) – Infoteria Corporation has announced the availability of Infoteria Business Language Objects for XML, or BLOX. Based on Infoteria's Asteria B2B platform technology and in-market experience, BLOX offers technology companies the ability to add the B2Bi capabilities of RosettaNet business processes to their J2EE or .NET Web services infrastructure.

Infoteria's J2EE Web services for RosettaNet, the first set of Infoteria's BLOX, are J2EE objects that run on any J2EE application server. They augment any back-end system by providing RosettaNet Implementation Framework (RNIF) support and Partner Interface Process (PIP)

message handling, including support for over 140 RosettaNet PIPs. Infoteria's BLOX for Microsoft's .NET Web services environment will be released early this year.

www.infoteria.com



DataMirror Announces DB/XML Transform 2.5



(Toronto) – DataMirror Corporation has released DB/XML Transform 2.5. The new version enables EDI-based firms to leverage existing systems and communicate with XML.

DB/XML Transform now supports RosettaNet, and includes



engine enhancements that enable it to perform faster and offer greater flexibility in creating mapping rules with more fine-grained transaction control for any-to-database transformations. The product also features more built-in functions and enhancements to its graphical user interface.

www.datamirror.com

XML NEWS


AltoWeb Application Platform 2.5 Available

(Palo Alto, CA) – AltoWeb, Inc., has added major features to its latest application platform. They include support for leading application servers, Web services support, built-in XML integration, support for custom connectivity, integrated team development,  model-based application testing, remote version-based deployment, and remote application monitoring.

AltoWeb Release 2.5 beta program customers and partners realized a tenfold increase in productivity and a reduction in time-to-market from months to weeks. A free downloadable 30-day evaluation copy is available at www.altoweb.com.



Consortium to Develop Background Check Standards

(Raleigh, NC) – The HR-XML Consortium has established a work group to develop a standard for background checks. The group will focus on developing an XML schema to support employer requests to third-party providers of background checking services.

The schema will be developed to support a variety of background checks, including those relating to criminal records, DMV records, education, employment history, and credit worthiness. The specification will support the specific requirements of HR-XML's Staffing Industry Data Exchange Standards (SIDES) initiative, which  includes a background check component. www.hr-xml.org


Bowstreet CTO Announced as Advisor to NeoCore

(Colorado Springs, CO) – Andy Roberts, chief technology officer of Web services leader Bowstreet, has been made an advisor to NeoCore. At Bowstreet Roberts


provides the company's technical vision and participates in the design of  Bowstreet's technology and products, including their XML initiatives. Roberts was instrumental in helping NeoCore adopt an XML-based product strategy. 

www.neocore.com

OASIS Committee Completes Validation Spec


(Boston) – OASIS has released RELAX NG 1.0 as a Committee Specification for validating XML-based languages. In support of the new specification, the  OASIS technical committee also released the RELAX NG tutorial and RELAX NG DTD compatibility. www.oasis-open.org

Core Networks Updates Flagship Product

(Halifax, Nova Scotia) – Core Networks Inc., a leading developer of activation and network management solutions for the cable broadband industry, has released CoreOS 4.0, the newest version of its flagship product. 

This latest release offers DOCSIS 1.1 and Oracle database support, enhanced network intelligence, billing and mediation integration, security enhancements, and proprietary modem support. www.CoreNetworks.com

Profile Systems Releases MasterDepot Version 4.0

(West Springfield, MA) – MasterDepot Version 4.0, a Web-based product data management solution that allows manufacturers and other selling organizations to manage clean, current, and complete product data files and exchange them securely in multiple electronic formats, is the latest release by Profile Systems. 


Native XML Database Supporting XQuery Launched

(Vancouver) – XML Global Technologies, Inc., has released GoXML DB 2.0, the first native XML database technology that provides significant support for the current working drafts of XQuery, an XML-based querying standard under development at the World Wide Web Consortium. 

New features include extensions for document insert, replace, and delete; full database security model; schema management; full text searching; automatic query optimization; integration with GoXML Transform; XML Spy integration. www.xmlglobal.com

Enhancements include a new data import tool, improved user interface, and extended data repository. The product also has new data export formats for two major electronic commerce initiatives – the e-Commerce Council of Canada's ECCNet multi-industry product catalog and the electrical industry's IDEA industry data warehouse. www.profilesystems.com

Documentum to Acquire Bulldog Technology

(Pleasanton, CA) – Documentum has signed a definitive agreement to acquire the Digital Asset Management technology of the privately held Bulldog Group. 

The addition of this technology will provide Documentum with a comprehensive, cost-effective ECM solution, offering more functionality and faster deployment than the multivendor solutions currently available. www.documentum.com


XML Global Introduces New Platform, New Program

(Vancouver) – XML Global Technologies, Inc., has released GoXML Transform Central, an extended e-business transaction platform that provides dynamic transformation solutions in Web service environments.

GoXML Transform Central also offers near-"frictionless" data exchange. It can be executed remotely, requires no client software, and allows automated translation of trading partners' business documents.

Concurrent with the release of GoXML Central is the launch of XML Global's ebXML Jumpstart program (www.xmlglobal.com/soln/jumpstart.jsp). The consulting service focuses on implementing EDI and XML business transaction systems. www.xmlglobal.com

DMSi, Software AG Release directWeb Portal Plus

(Reston, VA / North Andover, MA) – eidon GmbH Document Management Solutions, Inc. (DMSi), and Software AG have announced the release of  directWeb Portal Plus, an integration between the components of the eidonXportal family and Software AG's native XML database, Tamino.

directWebPortal Plus provides a cost-effective solution to the problem of delivering dynamically assembled XML data from any content management system via the World Wide Web. www.dmsi-world.com www.softwareag.com 



I Am an Undefeatable Hipster...

...and nothing can slow me down!



The Office Ninja has programmed for Cool Co Dot Com for two years. Most of the office had, at one time or another, complained about his Razor scooter. Despite that, he soon bought one for everyone in the office, to make the employees "more efficient and faster." Soon after, he also bought them Nerf guns, Nintendo consoles, and steering wheel joysticks.

BY THE SCOOTER-RIDING, NERF GUN-WIELDING OFFICE NINJA

No one can beat my kung fu style! I don't care if you're a Java programmer, Orc, or both. I'll defeat you either way. The rest of the office is in fear of my l33t programming skills, and my constant Nerf dart gun fights in the office. But I think most of the fear is from the former.

I mean, look at this page I made with XML! No one has 5kills like this! Dedicated to American Ninjas, it's the only completely dynamic Web page around where all the illustrations are pictures made from ASCII symbols that are generated by XML databases and XSLT stylesheets. It makes the site all text, and it loads slowly, but true programmers will see the beauty in such a site.

Sure, I don't understand every last programming request I get from the higher-ups, but I usually get the job done one way or the other. Sure, I don't quite "get" all the programming terms they give me, but I get the gist of them. The point is that I do my job, and I have the most fun doing it. When my co-workers and I get into our Nintendo races, or use the offices as our urban Nerf gun combat zone, we may cause a ruckus, sure. But it really raises employee spirits to see that much fun being had. I mean, right?

Well, most of my office playmates have been laid off in the past year. But I don't think that's indicative of the way all companies

treat fun-loving employees. I think it's just the economy and our uptight management.

The managers have always been hateful about my office happiness. They can't tolerate the fact that I actually enjoy coming to work. I have some fun in the office once in a while. Big deal. Well, it is to them.

Whenever I walk into an elevator full of all the stuck-up managers with their stuffy suits and ties, they see me with my ultra-cool Japanimation T-shirts, baggy pants, and skateboard (which, sure, I may be riding at the time). I see the envy burning in their eyes. Soon after one such encounter, they made up anti-skateboarding laws for the office. I couldn't believe it. Outlawing something as harmless as that out of spite. And the monkey suits didn't stop there. They started killing our freedom of speech.

Whenever the other programmers and I e-mailed each other to offer up an l33t match of programming skill, we cc'ed our messages to the rest of the office. I thought it would help office morale to see such gung ho workers. But no, the Nazi-esque sales team told us to knock it off. Sure, they e-mail the entire company every time they make a sale. But us, we can't tell anyone how great our new front-end Java servlet templates will be. Fascists.

Despite all the adversities that come my way, though, I persevere. My professional lifestyle will carry on here even if I don't. I didn't put the *Property of Cool Co Dot Com* masking tape labels on all my Nerf guns for the heck of it. If I am to depart this office someday, my legacy will live on with the new breed of programmers. They will be armed with my weapons of choice. To all hip, young programmers who would be disheartened I say Look up! Our day will come again. Well, unless none of the fun-loving programmers (who get laid off) find new jobs. But hey, everyone still wants employees like us. I mean, right? ☹



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

Software AG

www.softwareagusa.com/XML4U

Altova

www.altova.com