XML JOURNAL

The World's Leading XML Resource

Volume: 3 Issue: 2

XML-JOURNAL.COM

SYS-CON MEDIA

Sonic Software

www.sonicsoftware.com

Macromedia

www.macromedia.com/downloads

XML JOURNAL

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, Jean Cassidy
ONLINE EDITOR: Lin Goetz

WRITERS IN THIS ISSUE

Ravi Akireddy, John R. Callahan, Tod Emko, Jim Gabriel, Trace Galloway, Graham Glass, Roy Hoobler, Coco Jaenicke, Jim Milbery, Neil Raybould, Ajit Sagar, Hitesh Seth, Shouki Souri, Nick Xidis, John Zukowski

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $6.99/issue
Domestic: $77.99/yr (12 issues)
Canada/Mexico: $99.99/yr
all other countries $129.99/yr
(U.S. Banks or Money Orders)

EDITORIAL OFFICES

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

WRITTEN BY AJIT SAGAR EDITOR-IN-CHIEF

It's Still Rock and Roll to Me

Last year marked another significant period in the evolution of new technologies. Some of these evolutions reflect the various changes that we've seen in the economy. We saw Web services gain momentum, but skepticism still looms regarding the business models to which they'll eventually apply. XML has been firmly steeped in the very essence of most forms of data exchange via the Internet. At the same time, traditional mechanisms of data storage and exchange, namely the ERP and the RDBMS technologies, still control the real business tier.

An emerging trend indicates a return to plain Web technologies, as opposed to the development of a thick middle tier that would host the business logic for distributed marketplaces. The economy and failed business models of the past couple of years are driving some of these decisions, as the cost of developing the middle tier is high compared to that of developing applications that use plain old Web server services.

On one of my current projects we're dealing with data transformation using XSLT to generate customer catalogs. After going through the painful process of generating several instances of these catalogs, I find myself dealing with another application that strives to use XML in a situation that doesn't warrant it. XSLT is great for transforming documents; however, if these transformations involve the application of several complex business rules to generate the final output, then XPATH and XSLT probably aren't the best tools for the job. Once again we find ourselves with a hammer, looking for a nail.

Once upon a time, when RDBMS ruled the world, we used to store our data in flat tables. Then along came object-oriented languages and a new way of modeling the real world. Soon came object-oriented databases that highlighted the inefficiencies of converting objects to flat tables and then back to objects again. "Why not store data in a native, intermediate format?" developers asked.

For a short time it seemed that the OO vendors were going to provide the next incarnation of data storage. As we've seen, this initiative didn't last long. In the meantime, XML came along as a new, soon-to-be-ubiquitous technology for formatting data. In one of my previous editorials I tagged XML as the saving grace for OODBMS vendors. If you look at the OODBMS vendors of yesteryear (eXcelon, for example), you'll see that they're all XML server vendors.

However, while XML provides an excellent mechanism to format and transform data, it doesn't offer the best form of storage. Several XML vendors with native XML "databases" are currently trying to offer nonrelational storage for data. This is a viable alternative for some forms of data, such as contracts and purchase orders; however, this will never serve as the ultimate storage. In addition, as compared to SQL, XPATH is very tedious and limited for querying the data. Because the majority of the world's data is still stored in RDBMS systems and XML has become the accepted standard for expressing intermediate data formats, most of the major database vendors offer APIs for extracting data from RDBMS storage into XML documents. Finally, a lot of data schemas are still best suited to a relational model.

As the e-marketplace dream of auctions and exchanges has vaporized, the middle tier threatens to be thinner than expected. The majority of business logic is still driven by ERP systems. Again, ERP vendors are providing APIs for extracting documents into XML for intermediate storage and transformation. And, due to recent developments in technology, ERP vendors are exposing a part of their functionality as Web services.

After the last B2B fiasco, it seems that the tiers of a distributed system remain the same. The RDBMS systems own the data. The ERPs own the business processes. And newcomers XML and Web services have settled into their niches – they are the technologies that complete the picture.

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.*

HitSoftware

www.hitsw.com

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

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

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
Jean Cassidy jean@sys-con.com
Online 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 richarda@sys-con.com
Show Assistant
Niki Panagopoulos niki@sys-con.com
Registration Assistant
Jaclyn Redmond jaclyn@sys-con.com

CUSTOMER RELATIONS / JDJ STORE
Manager, Customer Relations / JDJ 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 GRAHAM GLASS]

Service-Oriented Architectures

While the short-term payoff of Web services technologies like SOAP and WSDL is faster, cheaper, simpler integration, I believe the long-term benefits will be more profound. Specifically, I think that Web services will catalyze a trend toward service-oriented architectures in which enterprises view their systems as an orchestrated web of services that extend beyond their firewalls into the systems of their partners and their customers. As I'll explain shortly, service-oriented architectures will require a new kind of distributed computing platform – in a sense, an operating system for Web services – that provides features like clustering, load balancing, fault tolerance, replication, and security in a way that is independent of the services themselves.

Until recently, the conventional wisdom was that an enterprise should strive toward an architecture based on a server-side component model such as CORBA, EJB, or DCOM. However, it's rare to find an organization where this is feasible. Different departments often prefer one technology to another, and even after bloody political wars in which one side finally dominates, it only takes a merger or acquisition to start the whole process over again. The good news is that Web services standards provide a common denominator for describing, advertising, and invoking services, thereby allowing enterprises to move toward a unifying service-oriented architecture that accommodates a wide spectrum of existing and future software technologies.

The fundamental principle behind service-oriented architectures is that every part of the software system is designed and implemented as a Web service, described using WSDL, and invoked using SOAP. Low-level services are orchestrated by higher-level services, and are typically located dynamically using a Web services broker to avoid hard wiring service locations into an application. Because WSDL and SOAP are universal standards supported by every software vendor, the implementation of individual services can be changed over time without any impact on the systems architecture. For example, a specific Web service could be initially implemented using Perl and then replaced by its .NET equivalent, and the architecture would remain the same.

Deployment of large-scale systems based on Web services requires a superset of the distributed computing technologies already found in application servers, including brokering, load balancing, clustering, and fault tolerance. And, more important, they must be implemented in a way that is independent of the Web services themselves. For example, if a Web service written in Perl unexpectedly goes offline, the system should be able to transparently fail over to a compatible service written in C# or Java.

On the surface, building a über-platform for service-oriented computing seems straightforward. Services can be expressed using WSDL and advertised using a federated version of UDDI. When a service is required, a UDDI request for a particular WSDL pattern is made and a service is selected from the matches that are returned. If a failure occurs, the system performs an automatic rebind by repeating the same procedure. However, numerous challenges have to be faced, such as how to unify the various standards for security and transactions, as well as how to provide developer APIs that hide the complexity of the underlying infrastructure.

Numerous projects are underway to implement a platform that supports service-oriented architectures. Sun's Jini was an early version of a service-oriented platform, but oriented around Java instead of Web services. In addition, its JXTA project is investigating P2P technologies that could be used to support a large-scale services platform. Talking Blocks is a start-up company that's working on an operating system for Web services, and my own company, The Mind Electric, is working on a platform called *GAIA* (an ancient term for a spirit that covers a planet). Other projects in the works will undoubtedly surface over the coming year.

I'm very interested to see how these new technologies affect the future of application servers. Since technologies for service-oriented architectures are a superset of those used within local clusters, they may replace the bulk of the proprietary infrastructure hosted within app servers. A similar thing occurred during the introduction of modern power stations, when much of the complexity of power generation moved from individual businesses to the power grid.

The realization of a platform for service-oriented computing is close at hand, and it's a tribute to the power of Web services.

GRAHAM@THEMINDELECTRIC.COM

AUTHOR BIO

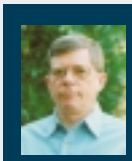
Graham Glass is CEO, chief architect, and founder of The Mind Electric. He holds a BS in mathematics and computer science from the University of Southampton and an MS in computer science from the University of Texas at Dallas.

UBL 'Might Help' .NET

So says Jon Bosak of Sun Microsystems, a founding father of XML, in an exclusive interview – keeping you in touch with the *i*-technologists that matter

Currently chair of the OASIS UBL Technical Committee, Jon Bosak *organized and led the working group that created XML, subsequently serving for two years as chair of the XML Coordination Group of the W3C. At Sun Bosak originated the strategy used for the Web distribution of documentation about the Solaris operating system.*

Interview by Jeremy Geelan, Editorial Director, ***XML-Journal*** • jeremy@sys-con.com



If anyone ever forgets that XML is not a language but a metalanguage, or thinks that it's new (it's not – it builds on 30 years of research and 14 years of SGML standardization), or doesn't realize that all XML languages can be processed by a single lightweight parser, or fails to see that its lack of limits on namespace or structural depth makes it powerful for data modeling...then Jon Bosak is definitely the man to set them straight.

Officially titled Sun's "Distinguished Engineer," the highly articulate Bosak has been Sun Microsystems' point man involved with XML ever since a cross-industry group, organized and led by Sun, first drafted it as a simplified subset of SGML capable of supporting the definition of an unlimited number of special-purpose languages optimized for different specific industries and domains.

The rest, as they say, is history.

SYS-CON Media's monthly online ***XML-J Industry Newsletter*** invited the team-spirited metalinguist to bring its audience up to speed on the latest e-commerce initiative he's spearheading: the Universal Business Language (UBL). Here ***XML-J*** presents excerpts of the exclusive interview, which appears in full at www.sys-con.com/xml/newsletter1.cfm.

XML-J: ***The inaugural meeting of the OASIS technical committee devoted to UBL was held last November in Menlo Park, California, and hosted by Sun. How would you briefly summarize Sun's specific interest in UBL?***

Bosak: At this point Sun's interest is just what it was when we organized and led the creation of XML – to do the right thing for the industry. Later on I hope we can leverage that leadership into products.

XML-J: ***Who are some of the other leading industry players involved?***

Bosak: That's a tricky question to answer because of the way OASIS technical committees are set up. The members of OASIS TCs are individual experts – they don't formally represent the companies they work for. So you have to be careful about attributing company sponsorship just because an employee is participating. On the other hand, subsidizing the work of a technical expert is a pretty big deal in this economy, so company affiliation isn't meaningless either.

Based on the resources they've committed to the work, the biggest corporate UBL supporters at this point are Sun, SAP, and Commerce One. At the first meeting we also got people from Boeing, VerticalNet, KPMG, GE, Contivo, Schemantix, the U.S. Navy, Oracle, HP, the U.S. Government Services Administration, Sterling Commerce, Pointgain, SoftQuad, AccessVia, LMI, and CSW Informatics. And we've got about a dozen other people from consultancies you don't ordinarily hear about. This is actually a good thing, because it means we're getting the really knowledgeable people that the big companies will be looking to for advice a year from now.

We've also got formal liaisons from some key industry groups – EIDX for the electronics industry, ARTS for retail, XBRL for accounting, and RosettaNet for information technology. I expect quite a few more of these vertical industry organizations to establish relationships with UBL as they start to realize that we're solving some basic information exchange problems that are common to all of them. And we're working on getting liaisons from the main EDI standards bodies, though that takes a while.

XML-J: ***Were you all worried by the sheer proliferation of markup languages?***

Bosak: Well, the whole point of XML is to allow the creation of an unlimited number of special-purpose markup languages optimized for different domains. And in fact, the OASIS process is set up the way it is in order to allow an unlimited number of special-purpose XML languages to be standardized in parallel. Some features of the process make sense only in that context. So in itself the proliferation of markup languages isn't a bug, it's a feature. It's basic to the whole XML concept.

What isn't basic to the XML concept is that you would have a heap of redundant markup languages for doing exactly the same thing. That's not helpful at all. Once you've identified a particular problem space, you want to assemble the people who know about that space and define one common markup language to deal with it. Without a general agreement on one way of expressing something, you can't really communicate. That's what language is all about – it's a general agreement about how to express things.

This isn't to say that you're always going to see just one markup language come out of every problem space. Different people have different ideas about how to solve these problems. But you should be able to expect that over the course of time these different approaches will start to converge. That's what I think we're seeing with UBL. We've got about half a dozen XML business libraries, and now it's time for them to start to converge.

XML-J: ***What would be the effect on e-commerce if there were not to be an initiative like UBL?***

Bosak: It would either be a lot more expensive and confusing to do e-commerce due to the continuing need to support half a dozen ways of doing the same thing, or a convergence would be forced on us by some big company's assumption of a dominant role. Given enough time struggling with all the adapters and so on needed to deal with multiple languages, people are eventually going to welcome some big player coming in and dictating a solution. So eventually I think we're going to end up with a single XML business library anyway. The question is whether it's going to be developed right now in an open process by a recognized bunch of experts out in the light, where we can all see it and comment on it, or...in some back room by a bunch of people concerned mainly with meeting their own product requirements. If it's a proprietary design, it will probably end up full of all kinds of weird hooks that make sense only in terms of where this particular set of developers thought their product was headed at the moment they froze the spec.

I'm for the open process, myself. You get better specifications that way. And if you want a legally binding international standard for commercial documents, which is what we're aiming for in UBL, then I think that development out in the open is really the only way you can make that happen in a way that will be acceptable to people all over the world.

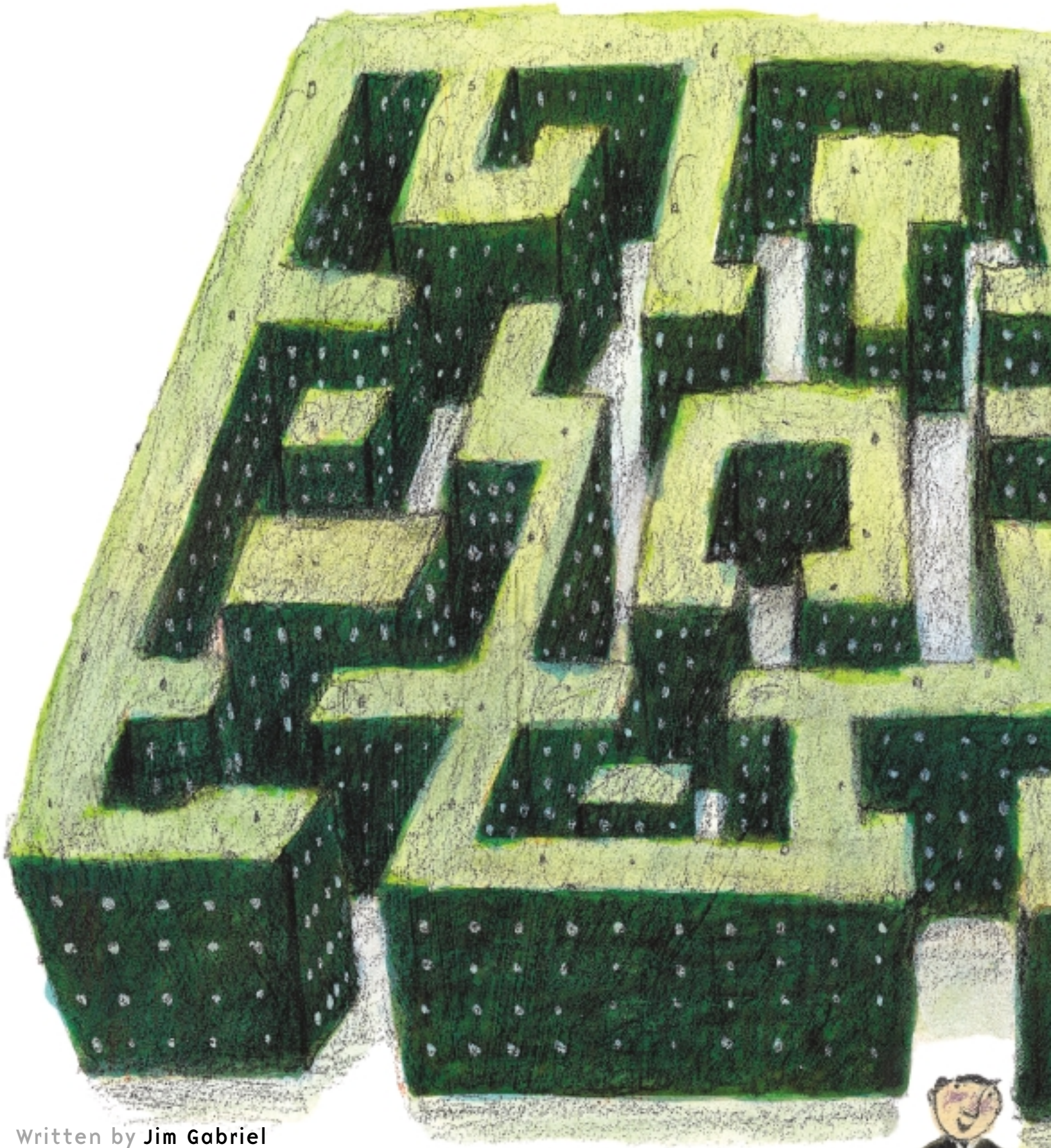
XML-J: ***Will UBL become extinct when ebXML finally arrives? And when might that actually be, do you think?***

Bosak: Let's take that one question at a time. First of all, UBL and ebXML complement each other; they're not competitors. So ebXML is not going to replace UBL. The deliverables promised for ebXML never included a designated XML syntax. The whole project was "syntax neutral" so that the semantic models could be specified in a way that leaves the binding to a specific notation undefined. This lets you produce XML or EDI versions of the data from the same models.

Continued on page 55

Data Mirror

www.datamirror.com/resourcecenter



Written by **Jim Gabriel**

Do we need 3GL-like XML development environments?

XML is object without source. There can be no development tools for XML until we find a way of creating source code.

DEVELOPMENT TOOLS FOR XML APPLICATIONS

"There's no such thing as an XML application." A strong statement, perhaps, but what do we mean when we talk about an XML application? Is a publishing system that relies on XML to do its work an XML application? Can we apply the term to a B2B marketplace where all the processes in a transaction can be defined by DTDs and all data flowing around the system is in XML? Is a content syndication system an XML application? Or a foreign exchange business line in an investment bank?

Typically, all of the above use XML extensively. But what do they have in common? And how do we "develop" these systems? Do we "develop" the XML parts, or do they fall out of the development of other parts? After all, XML isn't a programming language. XML applications are built in C++, Java, or Visual Basic (for example).

The Common Ingredient

The common ingredient in all XML applications is, of course, the XML. The data flowing around in the system is in an XML format. As such, I would argue that people don't currently develop XML applications. They develop software systems that use XML to achieve certain aims.

For example, XML is the glue in EAI (enterprise application integration), gluing together pieces of software that probably have nothing to do with XML. Publishing systems use XML to provide single-source to multiple-output formats, but the systems themselves aren't "developed" in XML, nor do they care very much how the XML has been defined. Publishing systems are usually built using a combination of technologies that can store, communicate, and manipulate XML, but that aren't intrinsically XML technologies.

Why Do We Need XML Application Development Tools?

Why do we care about XML application development? We care because putting an industrial-strength XML-based application together is complex and error-prone. The veterans of this industry will say that they do everything in vi or Notepad – and the implication is that development tools don't add much value, and perhaps can't even be trusted.

Explosion of Complexity

I disagree. XML allows you to develop sophisticated systems very rapidly, even if you do it all in vi (presuming you're working with a clean sheet). But once the system starts to take shape, you need help in the form of powerful software to build the XML pieces. This is because when you choose to use XML in your application, you're effectively taking single objects that are

important to your organization, such as a Transaction ID, and referring to them in dozens, hundreds, or even thousands of places by the time you've put the system together (see Figure 1).

This is denormalization to a dangerous degree. The Transaction ID will find its way into schema fragments, be passed to Java classes, be described in stylesheets, be transformed by XSLT, and so on. Clearly, unless you get it right the first time (and I assure you that never happens), you need development tools to help manage the complexity.

Adding Value

Let's examine the value that XML development tools might add. Is it time-to-market? Certainly. If a software program can fill in the missing pieces as you do your work, if a tool can catch errors, or if a tool can store pieces of work for reuse, you save time. Compared with doing everything in vi or Notepad, using a development tool is like using a word processor instead of a typewriter.

Another value-add is the ability to present a visual representation of something that's obscure in a clearer context, such as a file full of XSLT declarations. If you can present a source schema on one side of the screen, a result schema on the other, and in the middle display all the links relating source elements and attributes to result elements and attributes, you have a powerful representation of what the resulting XSLT will achieve for you. (You also don't need to learn XSLT to an expert level before attempting some fairly complicated transformations.)

Subliminal Education

Such tools can be educational, which is another great way of adding value. Having mapped definitions from one side to another, a good system should be able to generate the resulting XSLT and show it to you in a window, which will rapidly and subliminally train an attentive user in the use of XSLT. Compared with doing everything in vi or Notepad, using a development tool to conceal and enable complexity is like using a CAD/CAM drawing package instead of pen and ink.

Surely these are all very good reasons for using XML development tools.

What Should a Development Environment for XML Contain?

A development environment for XML allows you to develop XML. This sounds logical, even simplistic, but ask yourself: "What is the XML that we're trying to develop?" After all, XML comprises many things, not all of them officially sanctioned by the W3C. XML relies on a large range of non-XML technologies in order to function (think of CSS, for example). Where does the true XML flavor start and the non-XML flavor stop?

Essential Ingredients

To "develop" XML, you need to develop or use any of the following:

- XML document instances
- Schemas, such as DTDs, Schematron, or XML Schemas (XSD)
- XSLT for transformations
- XSL-FO for prepress applications
- XLink and XPath expressions
- Various queries using any of a range of querying languages or grammars
- CSS stylesheets
- Scripts (Perl, Python, Omnimark, etc.)
- And so on…

Essential Tools

The "tools" that will help you develop or work with the above list are all essentially graphical user interfaces around the files and fragments that come out of the end of the development effort, and include:

- Schema editor
- XSLT editor
- Stylesheet editor
- Script development tools
- And so on…

Essential Infrastructure

The infrastructure that needs to exist before any of these can be worked with includes:

- Parser
- XSLT transformation engine
- Browser
- XML (document) editor
- And so on…

Essential Standards

The development environment must enable applications to work with accepted standards, such as DOM3 and SAX.

'Nice-to-Haves'

The "nice-to-haves" in this picture might include:

- Content management system
- System for data-binding your XML into Java classes
- Web application server
- Web services development tools
- And so on...

We're rapidly arriving at an extremely complex picture of the requirements for an XML development tool, yet the picture is murky. Is it actually possible to combine all of the above into one tool? And if so, would anybody buy it or use it? What are we going to develop with these tools?

IDE

Let's look at the tools currently on the market. The term *IDE* (integrated development environment) is used by many manufacturers in the XML community these days. Products such as XML Spy from Altova and TurboXML from TIBCO/Extensibility spring to mind. The term *environment* represents a collection of tools in this context.

We've been conditioned by such products to interpret XML development as a means of creating DTDs, documents, stylesheets, and transformations. Yet when I create a DTD, then write a document that validates against that DTD, then store that document in a content management system, have I developed XML? Arguably, yes. But to what end? The

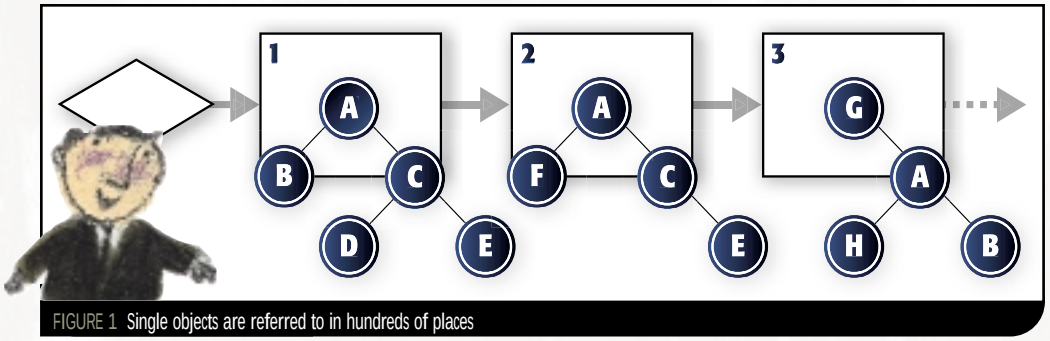


FIGURE 1 Single objects are referred to in hundreds of places

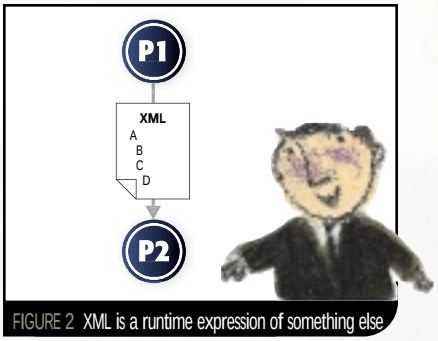


FIGURE 2 XML is a runtime expression of something else

BEA eWorld

www.bea.com/events/eworld/2002

XML does nothing without an infrastructure to use, publish, process, and communicate it.

Furthermore, as soon as I want to use an element in various places and for various purposes, I have a problem. Powerful software development environments provide the functionality to support true object reuse, inheritance, versioning, backups, and a permission system. The XML IDEs on the market at the moment don't.

If I create an XML-driven system that relies on a Transaction ID element (as described earlier), an XML IDE will help me put it into dozens of schema fragments, XSLT files, stylesheets, and so on. But the IDE won't manage the fact that the Transaction ID is actually one object that has been used in all these different places.

This is very disturbing from a software developer's perspective, especially if you're used to an object-oriented development environment and accustomed to the notions of, for example, single source and extension of object classes.

XML as a 'Language' Is Weird

The adoption of XML in IT is now unilateral, from finance to pharmaceutical, aerospace to defense. XML is mainstream. Nobody can afford downtime, and systems must support the business process flawlessly. In the pharmaceutical sector the legal repercussions of a mistake due to a problem in the publishing process could mean lawsuits totaling many millions of dollars. In the stock markets two minutes of downtime could represent millions of dollars in lost opportunity for an investment bank.

No Source Code

If we're going to work with XML in these environments, we should treat it with as much respect as all the other parts of the system. We should store the source code in a source control system, and apply versioning, branching, permissions, automated testing functions, and so on. But (I hear you say) XML doesn't lend itself to such management. There's no source code!

Compared with third-generation programming languages (3GL) such as C, COBOL, or Fortran, XML is very strange. Of course, comparing XML with 3GL isn't entirely fair. XML isn't a language in the traditional sense of programming languages. XML lets you use words that describe your business and apply constraints to those words. XML has no source code in the way that C++ or Java can be said to have source code, so we can't store and manage source code for XML. There are no XML compilers and development studios that provide the kind of functionality that, say, a Java developer would expect from an Enterprise version of Borland's JBuilder product.

However, it's this very inability to treat XML as we treat traditional programming languages that gives us so many management challenges in mission-critical environments. XML needs source, it needs powerful development environments and compiler-type technology, and it needs infrastructure to manage the development environment.

XML Is Impossible to Model

In complex IT environments, models are important. Models allow us to conceptualize the components in a system and generate the pieces that we need to develop software systems. For example, from a CASE (Computer Aided Software Engineering) tool or UML (Unified Modeling Language) modeling tool, I can generate Create Table scripts and stored procedures that build my database environment for me. Models simplify the work of application integrators and allow people who care about business processes to relate the functioning of the organization to the IT infrastructure supporting the organization.

XML Is a Runtime Expression of Something Else

As seen in Figure 2, when two processes exchange information in XML:

1. The XML represents a small part of a process model.
2. It's probably a subset of a database schema.
3. And it probably contains temporary values generated for the purpose of the current process but not likely to be stored.

The XML is structured according to a format that has been agreed on by the programmers or the designer or the supplier of some of the software involved (for example, CommerceOne). The model describing this exchange of data might be found in UML or a CASE tool, or be derived from a group of functions acting on data that can be described by a database schema. The model for the XML document passing between the two processes doesn't exist; it doesn't need to because well-formed XML obeys a set of rules that render the model redundant.

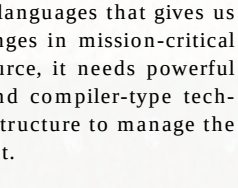
Serious Problems

At least, that's the idea. That is also the reason so many organizations are beginning to run into serious problems with XML development. The following problems arise for the developer:

- Implicit validation means there's no means of enforcing the contract between P1 and P2 other than to return an error and fail when things go wrong.
- System documentation should exist to describe the contract between the two processes (and a formal syntax is preferred when describing XML).
- There's no "model" of the communication, therefore no way of understanding in a syntax-free way the XML part of the process between P1 and P2.
- Without a formal description of the communication, when one of the elements in the XML document is affected by a change elsewhere in the system, you have no way of knowing that the process between P1 and P2 is probably broken.

A Model of the Document Is Not a Solution

To solve the foregoing problems, we need to model the document in a schema or DTD. The "model" of an XML document is its schema (see Figure 3).



However, having a schema simply isn't sufficient. Think back to the Transaction ID object mentioned

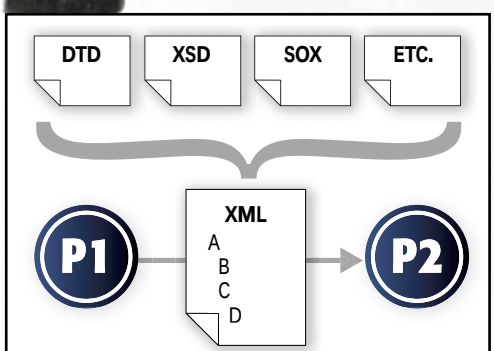


FIGURE 3 Schemas describe documents; they don't model them

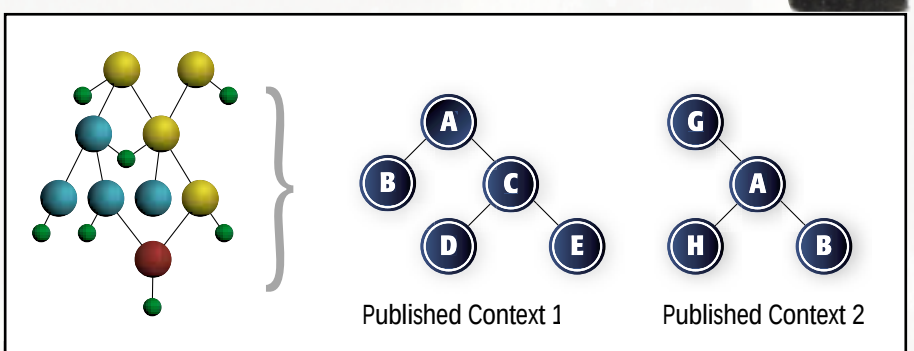


FIGURE 4 True object reuse and dependency tracking

> The Adoption of XML in IT is now unilateral, from finance to pharamaceutical, aerospace to defense. XML is now mainstream.

earlier. I care about transaction IDs because I'm building a software system for a commercial application – an insurance claims system, say, or a foreign exchange business line application for an investment bank. The Transaction ID isn't something that starts life in XML – it's probably there because of gigabytes of legacy data, existing software applications, and a UML model on somebody's desktop. It turns into XML because I'm using XML to glue all the different pieces of my system together with XML. And my transaction ID travels around and touches every system, every program, every database.…

My company acquires another company and we start to integrate our IT infrastructures. My transaction ID is an integer. The other company's transaction ID is an integer plus a timestamp. I've built my system using XML; my transaction ID occurs in thousands of different places and I have no way of knowing where. I don't have a model of my XML. The upside of XML when I built the system was the phenomenally short time-to-market for a highly sophisticated system. The downside is that the result is difficult to maintain and there are inherent risks.

A Document-centric Example

Imagine the publishing system at a pharmaceutical company. My project scope is the packet of tablets that a pharmacist sells to the public. I have four documents: a label on the drugs, a label on the box, a sheet of information inside the box for the customer, and a sheet of information for the doctor who prescribes the drugs. I have 300 products. I sell in 130 countries. I need to publish in at least 30 languages. I have to deal with the local legislation and cultural requirements of each country. XML offers me the chance to build the perfect solution (and nothing else can).

One of my attributes is a piece of text describing a certain warning in a certain set of circumstances for pregnant women at a certain stage in their pregnancy. By the time I've rolled out my product in 130 countries and 30 languages, a complex authoring and publishing environment will have been built to process the attribute, and the attribute itself will have been used or referred to in potentially millions of places. There's no way of knowing where, or what the impact will be if a related property needs to change. Not only is this dangerous (if an error in my publishing system causes a life to be lost, is this XML-induced death?), but you can't forecast budgets for unpredictable outgoings.

How can I manage the complexity of such a system? This is where a 3GL-like XML development environment could significantly reduce the risks and costs involved.

Creating Source Code for XML Environments

Let's examine what it would take to create a 3GL-like development environment for XML. First, we need source code. If a schema is neither the source code, nor the model, for an XML document, what else can play this role? The answer is that we need a way of modeling the model. In other words, we should create an object model that can store enough information about a structure to be able to generate the pieces needed in the XML-based application. It should be possible to generate schemas, stylesheets, and the like from an object model.

Enabling Object Reuse and Inheritance

An extra dimension is required for true object reuse and inheritance (see Figure 4). For example, to treat a transaction ID as one object that has been used in multiple places, we need a model of the model of the model of the document. That is, from an object model we need to be able to assemble structures (models of schemas) from which we can generate schemas and other property files, which describe the validation rules and process requirements of a document.

The structures we assemble are like contexts in which an object is used. In context one, *A* contains *B* followed by *C*, which contains *D* followed by *E*. In context two *G* contains *A*, which contains *H* followed by *B*. If I know that *A* in any particular context is actually a reused object from a model where it occurs as one and only one object, I can apply true object reuse and inheritance.

The next hurdle is versioning of objects. This is where the underlying technology becomes very important. In any sophisticated source control environment, developers can check out an object, work on it, and check it back in again. This activity is controlled by a versioning mechanism that allows updates to happen on a branch of the parent code line. Full permissions and transaction support needs to be applied. Multiple variants of a code line are possible. Published versions are identified by a time line that essentially picks up the relevant pieces and freezes them into a state that cannot be violated.

The only technology that can support this level of control over source code is repository technology. In other words, everything about the model of the model of the model of the document needs to be captured and managed in a database, which requires a very sophisticated approach to metadata. Get this piece of the puzzle right and XML development will take a quantum leap forward! To my (admittedly biased) mind, only one product anywhere in the world is capable of providing such support – CorteXML from Barbadosoft, which was designed to enable high-speed change in complex XML-based environments.

Conclusion

Currently, XML development environments do not provide the level of developer support customary in a 3GL programming environment. The question we need to answer is, Do we need 3GL-like XML development environments? For simple systems, probably not. But for complex systems I would argue that we cannot do without them.

Author Bio

Jim Gabriel is CEO and cofounder of Barbadosoft, an XML infrastructure software company founded to provide the infrastructure needed to enable high-speed change in complex XML-driven environments.

JIM@BARBADOSOFT.COM

[WRITTEN BY JOHN R. CALLAHAN

Preprocess requests and postprocess responses at a semantic level

Intelligent XML Content Firewalls

Current Web-based application servers can be complex and monolithic because they must address a myriad of issues, including business logic, security, privacy, data integration, presentation, session management, caching, legacy integration, and database access within an overburdened object model.

Component-based approaches (EJBs, etc.) help separate some of these issues, but a traditional firewall model can be used to filter, cache, and redirect digital content at a semantic level before and after processing by a back-end server. I introduce the notion of a content firewall (also known as a *semantic firewall*) to preprocess requests and postprocess responses at a semantic level using a series of intelligent XML transformations. By intelligent, I mean that a content firewall applies high-level rules to generate and apply XML transformations via dynamically generated XSLT. These rules can easily be expressed so information assurance policies can be quickly configured on a Web server, application server, or proxy server.

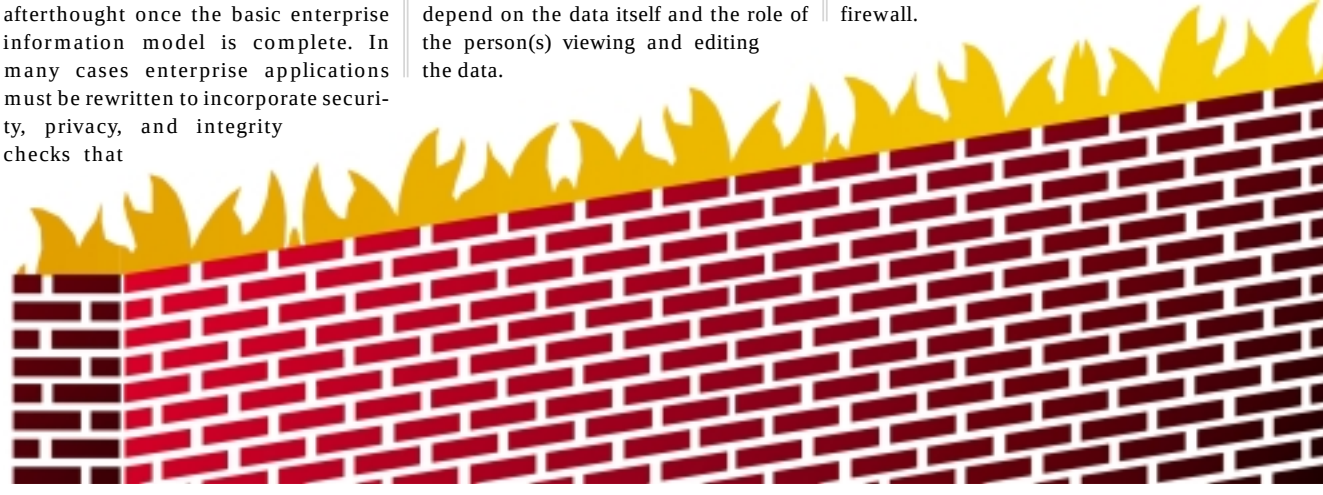
Many companies use XML dialects to encode their e-business information models, but fail to integrate the information assurance aspects of their business processes into these models. Information assurance is usually an afterthought once the basic enterprise information model is complete. In many cases enterprise applications must be rewritten to incorporate security, privacy, and integrity checks that are outside the scope of the core information model but entwined within the business process itself. Evolving information models and security policies exacerbate the problem by forcing designers to develop complex, intertwined solutions that aren't scalable and prove difficult to configure.

IP firewalls and site management tools provide important access control to URLs, files, and directories, but they're not designed to filter elements of the actual content flowing through them in either direction. For example, role-based access control (RBAC) and task-based access control (TBAC) are difficult to integrate into current site management tools because such control depends on understanding the semantics of the digital content itself, the business process within which the document flows, and the role (and task) of the currently authenticated user. Current packet-based and file-based access control models aren't powerful enough to manage access decisions that depend on the data itself and the role of the person(s) viewing and editing the data.

Content Firewalls

Semantic-based filtering of digital content is needed to decompose monolithic server applications into a series of XML transforms. These transforms can be connected to implement business processes. XML provides the necessary glue so that data can be self-describing between transforms.

A content firewall consists of a series of intelligent XML transforms that implement steps of a business process, perform enterprise application integration (EAI) tasks, enforce information assurance policies, audit content, or stylize content to different devices and users. Filters within a content firewall perform a series of filtering operations on XML content using XSLT transformations that are dynamically generated from high-level rules. XML Schemas (or DTDs) are used to define the syntax for XML content flowing between filters. This ensures that the output from one filter is compatible with the input expected by another filter in a transform series. High-level rules are used to perform semantic transformations on the XML content passing through each filter in the firewall.



AUTHOR BIO

John R. Callahan, CTO of Sphere Software Corporation (www.sphere.com), Columbia, MD, has worked for NASA, WVU, IBM, and Xerox Corp. in Palo Alto, CA. He holds a PhD in computer science from the University of Maryland, College Park. As an academic fugitive, John earns a living on the run as a teacher, consultant, software developer, and systems architect.

Each filter can add, delete, change, encrypt, and decrypt attributes, nodes, node sets, or tag contents. Filters within a content firewall can also be configured to log, audit, trace, and redirect incoming or outgoing XML content.

High performance is achieved through the use of caching, preparsing, prefetching, and, primarily, using a pipeline of SAX transformations. Transformations using SAX (Simple API for XML) can be pipelined together to form a chain of transformations that feed events efficiently down a chain of tag handlers. Similar to Unix pipes, these transforms are implemented as separate Java threads. This means that a thread doesn't have to wait until the previous thread in the pipeline completes.

A Simple Example

Figure 1 depicts a three-phase content firewall (response filters only) configured between a back-end database and Web clients using browsers. The database generates XML content in response to an incoming query request. In this example the content firewall comprises three filters that transform the XML content within an HTTP response from the back-end server (e.g., a Web-enabled database). The filters transform the outgoing XML content by adding, changing, deleting, and formatting the information for end users based on their authenticated role and high-level security rules.

Filter 1 simply attaches session information to the XML content. This information is then used by Filter 2 to decide which elements of the content can be viewed by the authenticated user making the query request. In this example, Filter 2 adds a view attribute to each record in the XML content indicating whether or not the authenticated user can view a record. Finally, Filter 3 transforms the entire XML result into XHTML by stylizing the XML response based on the view attribute added by Filter 2 to each record.

Let's step through this simple example by examining the XML transforms performed by each filter in the pipeline shown in Figure 1. In response to a SQL query such as:

```
select id,status from patient_tests
as xml
```

the back-end database generates an XML fragment as output. This output XML fragment (see Listing 1) from the query is then passed to Filter 1 as input. Filter 1 adds session context information (i.e., the name and role of the current authenticated user making the request) as attributes in the top-level tag. Filter 1 also transforms each record element by looking up the patient identifier in an LDAP database along with the name of the patient's doctor. The <id> tag is eliminated and the <name> and <doctor> tags are added to produce the XML fragment (see Listing 2) as output from Filter 1.

This fragment is then passed as input to Filter 2. Filter 2, the heart of this example content firewall, applies complex security rules in order to transform the input by adding, deleting, or changing tags, attributes, nodes, and node content. In this example Filter 2 adds a view attribute to each record to indicate which records should be shown or hidden by the next filter in the pipeline. It adds the view attribute to the status tag of each record based on two rules:

Rule 1: All physicians can see the list of patient records.

Rule 2: Only a patient's physician can view a test record.

Based on these rules, Filter 2 produces the XML fragment shown in Listing 3 as output. The status element of the first record (Smith's test result) can be viewed by the current user (Jones) because (1) Jones is a physician and can view all the records according to Rule 1, and (2) Jones is Smith's doctor (thus satisfying Rule 2). The view attribute of the status element of the second record (Morgan's test result) is marked "false" because even though Jones is a physician (thus satisfying Rule 1), the status can't be viewed by the current authenticated user according to Rule 2 (Zane is Morgan's doctor).

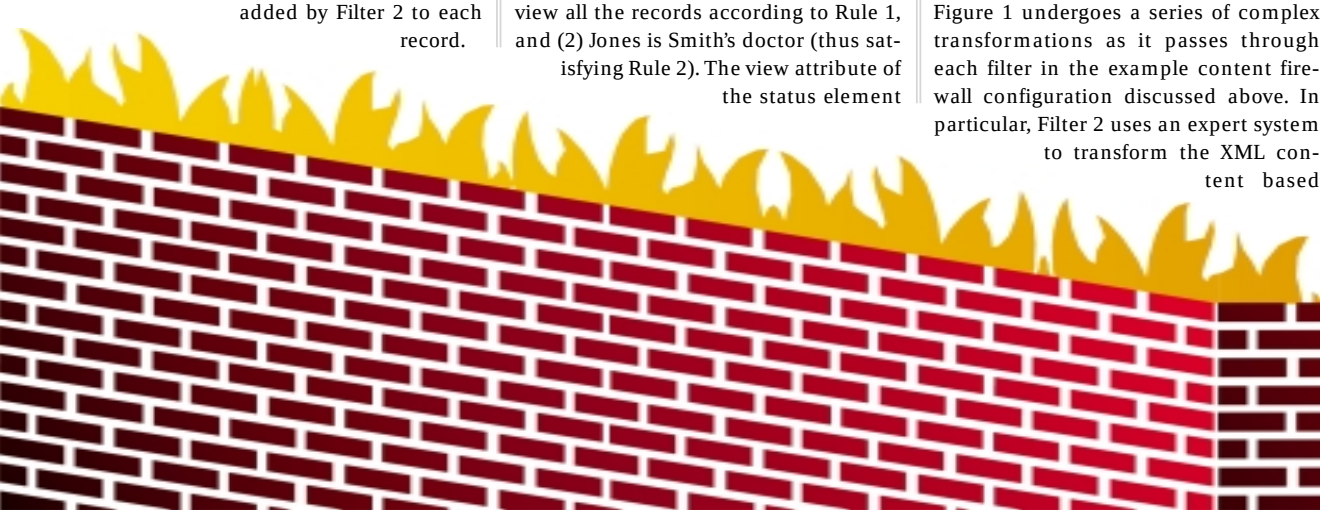
The XML fragment shown in Listing 3 is then passed as input to Filter 3, where it is stylized based on the XML content produced as output from Filter 2. Filter 3 uses an XSLT stylesheet to transform the XML content into XHTML. The XSLT stylesheet uses the role attribute of the top-level tag (e.g., physician) and the view attributes on the status elements to transform the XML content into the appropriate XHTML for presentation in a requesting client's browser. The XSLT templates shown in Listing 4 are part of an XSLT stylesheet used to transform the content as appropriate based on the view attribute of each record's status element.

The final XHTML produced as output from Filter 3 might look like the fragment shown in Listing 5. The resulting XHTML is then sent to the requesting client in the body of an HTTP response. Filter 3 is implemented within the content firewall, which is within a firewall server or implemented as a postprocessing filter on a Web server. The stylization doesn't occur within the user's browser in this case.

This example of a content firewall configuration (as shown in Figure 1) only illustrates the filtering of an outgoing XML response, but incoming XML documents and GET/POST variables can also be transformed by a series of filters within a content firewall. Elements of an HTTP GET or POST request (e.g., header and form elements) can easily be encoded within XML and filtered before query processing via intelligent filters.

Rule-Based Transformations

The XML content generated by the query on the back-end database in Figure 1 undergoes a series of complex transformations as it passes through each filter in the example content firewall configuration discussed above. In particular, Filter 2 uses an expert system to transform the XML content based



on high-level rules. These rules can easily be specified as conditions and actions on XPath attributes and node sets.

Rule 1 states that "all physicians can see the list of patient records." This rule can be specified as a transform based on conditions and actions expressed relative to the attributes and node sets corresponding to the XPath expressions for /results/@role and /results/record from Listing 2.

Given the XPath values for these expressions from Listing 2, the conditions for Rule 1 are satisfied if:

```
/result/@role equals the singleton
set {physician}
```

If this condition is satisfied, then the transformation action for Rule 1 is triggered:

```
set the view attribute to "true" for
each /result/record/status
```

This action sets the view attribute of *all* /result/record/status elements to "true" even though Jones is not Morgan's physician. But Rule 2 states that "only a patient's physician can view a test record." Rule 2 is applied after Rule 1 and can be specified as a transform based on conditions and actions on the attributes and node sets for the XPath expressions for /results/@user and /results/record/doctor from Listing 2.

Given the XPath values for these expressions from Listing 2, the conditions for Rule 2 are satisfied if:

```
/results/@user doesn't equal
/results/record/doctor
```

If this condition is satisfied, then the transformation action for Rule 2 is triggered:

```
set the view attribute to "false" on
the /results/record/doctor/../status
```

If Rules 1 and 2 are applied in succession for all applicable cases, the input XML content to Filter 2 is transformed appropriately and then piped to Filter 3, as discussed previously.

The primary advantage of the rule-based approach to XML transformations is that each rule can be specified independently of other rules, although the order of rule application is critical. Another advantage of the rule-based approach is that rule conditions and actions can be specified in a language (i.e., XPath sets) that's closer to the high-level goals and policies of a content provider. By expressing business semantics as rule-based transforms, enterprises can operationalize their policies without relying on proprietary business rule beans, low-level code, complex configuration files, or complex XSLT stylesheets to perform similar XML transformations.

To test a rule-based XML transformation process, sample documents are used to establish a benchmark so that the conditions and actions applied during a transform process can be traced in the order in which they're performed. This allows managers to audit the order of rule application (conditions and actions) and validate their compliance with corporate security and privacy policies.

Summary

IP firewalls and site access tools still play a vital role in enterprise security and content management. Site management solutions, virtual private networks, encryption, and packet firewalls were developed to relieve application programs from the burden of handling security management issues. Most application programs aren't concerned with whether or not specific IP addresses are disallowed, or a user is barred from login, or that only certain users can invoke a CGI. Indeed, these issues are typically handled using external configuration files and other site access management programs that can be dynamically reconfigured without service interruptions. Most of these configuration files and support programs can be managed by nonprogrammers with standard training and certification.

The content firewall model applies high-level rules to generate XML transformations dynamically. These rules can be configured by nonprogrammers so that information assurance policies can be expressed quickly, implemented, and audited outside the core business logic of an application server. Using a layered architecture, the performance of a content firewall can be optimized by pipelining the transformations, caching back-end data requests and responses, and pregenerating transformation templates.

JCALLAHAN@SPHERE.COM

LISTING 1

```
<?xml version="1.0"?>
<results>
 <record>
    <id>4207</id>
    <status>positive</status>
  </record>
  <record>
    <id>4208</id>
    <status>positive</status>
  </record>
  ...
</results>
```

LISTING 2

```
<?xml version="1.0"?>
<results user="Jones" role="physician">
  <record>
    <!-- id removed -->
    <name>Smith</name>
    <status>positive</status>
    <doctor>Jones</doctor>
  </record>
  <record>
    <!-- id removed -->
    <name>Morgan</name>
    <status>positive</status>
    <doctor>Zane</doctor>
  </record>
  ...
</results>
```

LISTING 3

```
<?xml version="1.0"?>
<results user="Jones" role="physician">
  <record>
    <name>Smith</name>
    <status view="true">positive</status>
    <doctor>Jones</doctor>
  </record>
  <record>
    <name>Morgan</name>
    <status view="false">positive</status>
    <doctor>Zane</doctor>
  </record>
  ...
</results>
```

LISTING 4

```
<xsl:template
match="record/status[view='true']">
<xsl:value-of select="."/>
</xsl:template>

<xsl:template
match="record/status[view!='true']">
unknown
</xsl:template>
```

LISTING 5

```
<p>
Patient test results for Dr.<br/>Jones:<br/>
<table border="1">
<tr><td>Smith</td><td>positive</td></tr>
<tr><td>Morgan</td><td>unknown</td></tr>
</table>
</p>
```

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

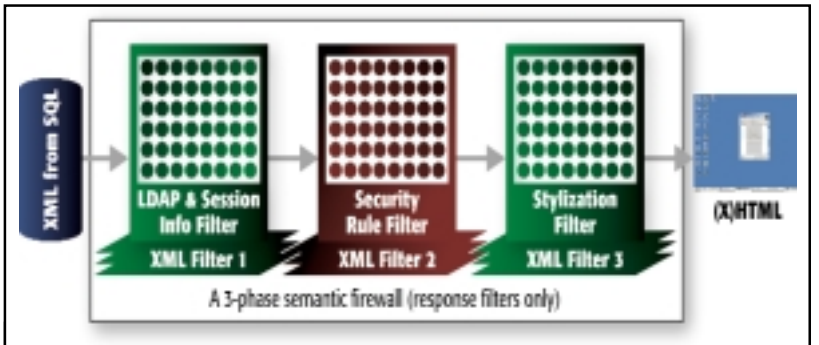


FIGURE 1 Example content firewall configuration

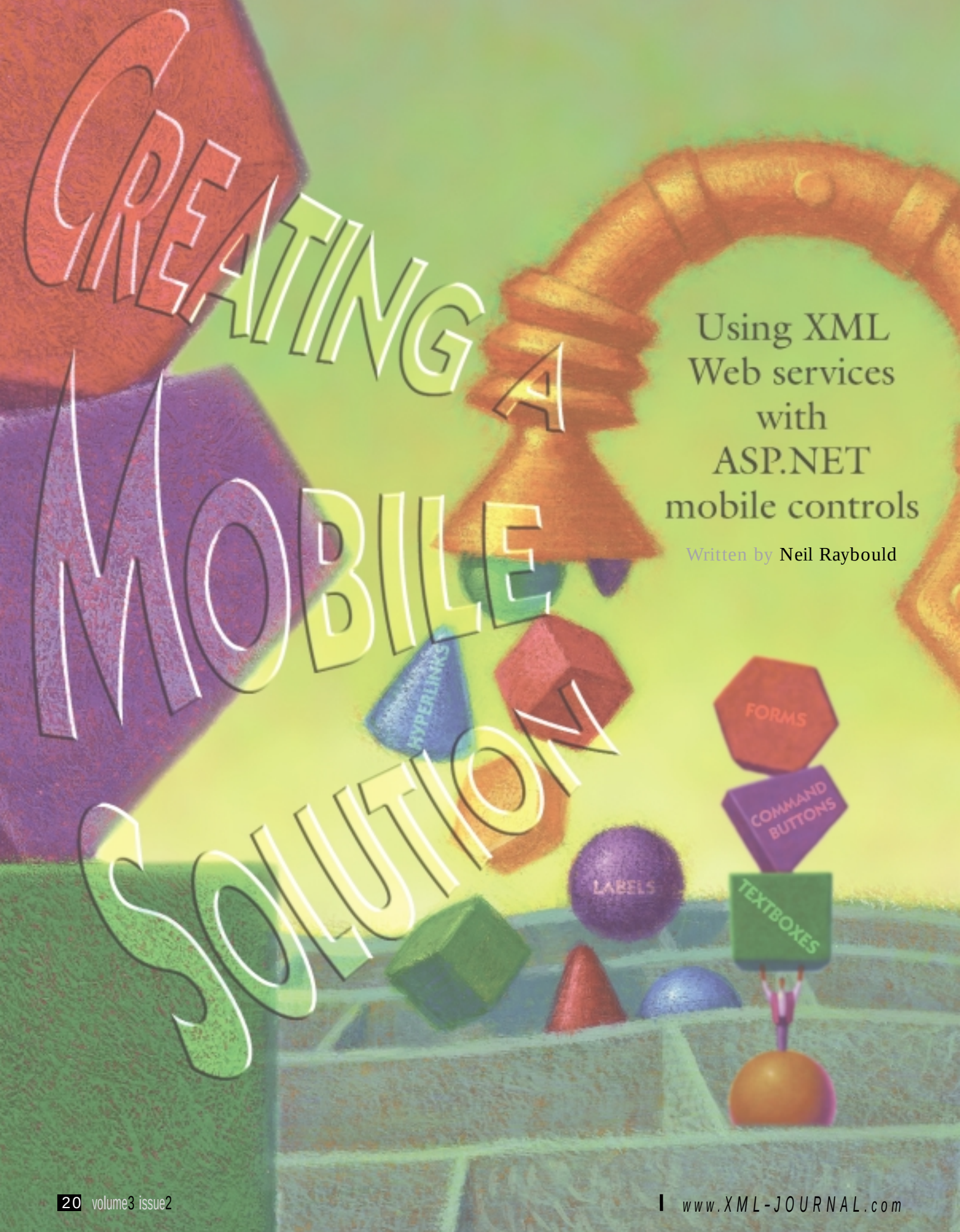
.NET Developer Conference & Expo

http://events.internet.com

Creating a Mobile Solution

Using XML Web services with ASP.NET mobile controls

Written by Neil Raybould



THERE'S BEEN A GREAT DEAL OF DISCUSSION ABOUT HOW WEB SERVICES WILL BE USED, WHICH IS TYPICAL AS ANY NEW TECHNOLOGY BEGINS TO EMERGE FROM ITS INFANCY. AN ESTIMATE BEING TOSSED ABOUT IS THAT, OF ALL OF THE WEB SERVICES EXPECTED TO BE CREATED, LESS THAN 1% CURRENTLY EXISTS. AS WE KNOW, A WEB SERVICE ISN'T MEANT TO LIVE PRIMARILY AS A STANDALONE COMPONENT, BUT AS PART OF A VIRTUAL APPLICATION OR SEVERAL APPLICATIONS.

THIS ARTICLE DEMONSTRATES ONE SUCH SCENARIO BY BUILDING A MOBILE APPLICATION USING A WEB SERVICE IN CONJUNCTION WITH AN INCREASINGLY POPULAR TECHNOLOGY: ASP.NET MOBILE CONTROLS.

I'm assuming that the reader has some familiarity with ASP.NET, specifically C#, VB.NET, and Web services. To ensure that the example runs properly, the Microsoft .NET Framework and the Microsoft Mobile Internet Toolkit (MMIT) must be installed on the Web server. You must also have an established connection to the Internet.

Following is a brief introduction to ASP.NET mobile controls.

ASP.NET Mobile Controls

Simply put, ASP.NET mobile controls are Web-based elements such as forms, labels, textboxes, command buttons, or hyperlinks that are rendered on a mobile device (or a PC's Web browser). Mobile controls are Microsoft's answer to the need to create mobile solutions for a variety of supported devices without having to write code for each.

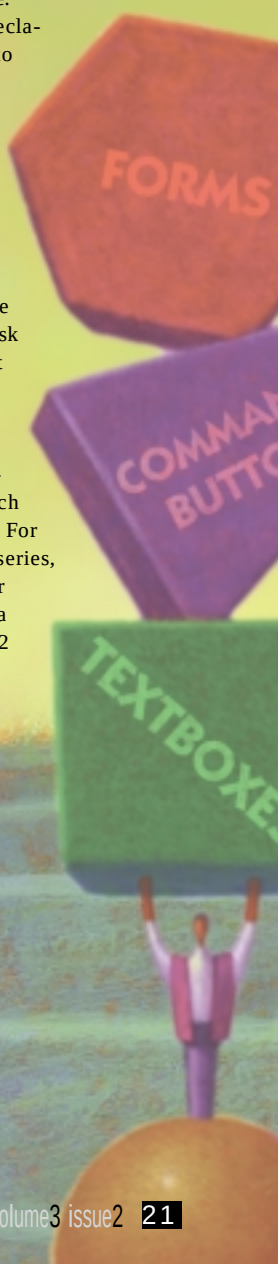
To enable the .NET Framework to render the appropriate output for the device dynamically, the MMIT mentioned above is installed as well. It works in conjunction with the .NET Framework so that once the Internet Information Server (IIS) receives HTTP requests, content can be created for mobile screens on any number of .NET-supported devices (based on the HTTP USER AGENT server variable in the HTTP header). Once a match is made with the device type, the Web server then compiles the page using an appropriate converter and responds with the resulting content that device needs. Some of the devices currently supported by .NET are the Pocket PC, Mitsubishi T250, Nokia 7110 and 6210, Sprint Touchpoint, Samsung Touchpoint, and Microsoft Internet Explorer 5.5 or greater, with promises of increased support for additional devices in the coming year.

A Simple Mobile Page

Development begins with a single mobile page, which is a file with an ASPX filename extension that exists on a .NET Web server. One or more mobile forms can exist within a mobile page, with multiple mobile controls contained within each mobile form. Having all the display code located in one file helps improve application performance.

We must first include the .NET compiler declarations in Listing 1 at the top of the mobile page to make use of mobile controls. As you'll notice, I've indicated that server-side script will appear in C#. After these two statements, all mobile controls in the mobile page contain the "mobile" prefix created in the register directive, followed by the name of the element. For example, to ultimately have access to two mobile forms, each containing a textbox with text, we can use the code in Listing 2.

As I mentioned earlier, we can have multiple mobile forms within one page. This may lead you to ask how the device knows which form to display. Without any intervention the first mobile form in the series will be the screen that appears by default on the device. In the example Form1 will appear first. However, a property called ActiveForm can be set programmatically to let the mobile page determine which mobile form to display based on a given action. For instance, since Form1 is the initial mobile form in the series, it will appear first. Yet in Listing 3 the onclick event for the mobile:Command element, Button1, contains a routine that sets the ActiveForm property to Form2 when it's clicked, thus displaying Form2.



To test this out yourself, create the directory c:\inetpub\wwwroot\XMLJournal. Now create a file called MobileTest.aspx within the XMLJournal directory, and enter the three code segments (Listings 1–3 in succession) within this file. To see it work, we can use Internet Explorer to browse to the page (http://localhost/XMLJournal/MobileTest.aspx). It'll be rendered in HTML as if it were a regular Web page.

The advantages of ASP.NET mobile controls become clear in that we can also view this mobile page in the Microsoft Mobile Explorer (MME) emulator (more about this below). Figure 1 shows how this is displayed in comparison from that of Internet Explorer output.

Since MME requires WML content, the output sent from the Web server will be different from that of Internet Explorer. The only major difference in how it's rendered is that the button appears as a link. This element conversion is performed by the Web server.

With this brief understanding of how ASP.NET mobile controls operate, let's now look at our featured mobile solution, Sojourner.

Building the Sojourner Application

Sailors, pilots, hikers – they all want to have weather information on their particular location available at any given time. With the increasing pervasiveness of wireless Web-enabled devices, mobile users can receive information about their location by entering the postal code or city/country combination.

We'll address this need by creating a tool called *Sojourner* that contains a mobile interface for selecting these values and getting information about a particular area. Figure 2 shows the solution architecture. We'll look at AreaWeatherWS, an ASP.NET Web service in VB.NET that will retrieve specific information from a weather site based on a postal code and return the weather. Finally, the mobile page, which is written in C#, handles presentation logic, calling this Web service as needed and rendering its results for the given mobile device.

To run the code on www.sys-con.com yourself, we assume that you're using http://localhost/Sojourner as the virtual directory, which corresponds to c:\inetpub\wwwroot\Sojourner. Within this directory another folder called *bin* must be created.

Sojourner Web Service

This Web service is written in VB.NET to showcase the .NET Framework's ability to handle different languages. The Web service uses a technique known as *screen scraping*. We can see the code in Listing 4, which is saved as AreaWeatherWS.asmx in c:\inetpub\wwwroot\Sojourner.

There are a couple of things to note. In VB.NET we make the various functions available for consumption within the Web service by using a <WebMethod> attribute before the routine name. This particular WebMethod contains a function called *GetWeather*, which accepts a parameter called *query*. When this function is executed, an HTTP request is invoked to the weather Web site using that parameter. Based on the HTML generated, it uses regular expressions to parse out the text needed for our display, which is contained in the class GetWeatherMatch. This class contains the string Weather, which is returned by the Web service. Once this code has been entered, we can test it as described below.

FIGURE 1 Expolrer output vs MME output

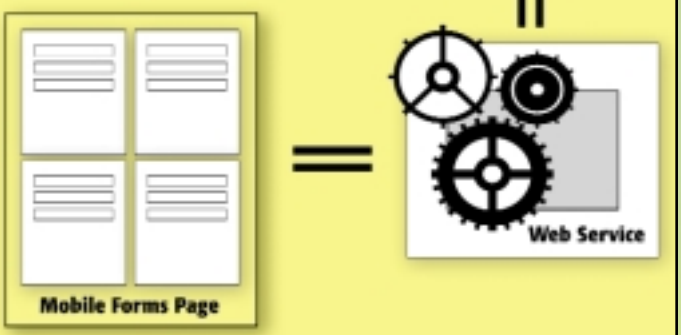


FIGURE 2 Sojourner components

Using the WSDL Test Page

Using Internet Explorer, we can see the Web Service Description Language test page by entering the Web service's URI in the address window (e.g., http://localhost/Sojourner/AreaWeatherWS.asmx). This will display a screen showing each of the consumable WebMethods as hyperlinks. As we click on a WebMethod link, we can impersonate a consuming application by entering values as if they were being received from an application. We have only one method, GetWeather, so we click on it to see the next screen in Figure 3.

Figure 4 shows the XML result generated in a new browser window when a value is entered in the text field and the Invoke button is pressed. Since we specified the Namespace as http://localhost/Sojourner within the Web service code, we won't see references to tempuri.org. The result is returned in a long string called *Weather*. The element Weather is nested within the GetWeatherMatch class element. This is the class that was returned based on the regular expression result in the string Weather when we invoked the HTTP request to the site.

Once we're confident that the Web service is working correctly, we must create a proxy so that consuming applications can use it.

Creating the Web Service Proxy

After the code has been entered, we compile the Web service proxy using a couple of utilities, wsdl.exe and vbc.exe (since this is a Visual Basic.NET file). To shield us from entering the myriad of switches required for each, I created a batch file called *ProxyCompiler* that can either be double-clicked or run at the command line. The file sets the appropriate path and executes the compilation applications (see Listing 5). Since we're working within the c:\inetpub\wwwroot\Sojourner directory, ProxyCompiler can be placed in the Sojourner directory with the AreaWeatherWS.asmx file.

Before running this batch file, you should verify all directory paths to ensure that they correctly point to the appropriate executables on your

> "Simply put, ASP.NET mobile controls are Web-based elements that are rendered on a mobile device"

Missed an issue?

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

JAVA DEVELOPER'S JOURNAL

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

Check out over 500 articles covering topics such as...

Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, Libraries, Embedded Java, XML, Wireless, IDEs, and much more!

JDJ The Complete Works
~~Reg $119.99~~
Buy Online
Only $71.99

XML JOURNAL

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

Check out over 150 articles covering topics such as...

XML in Transit, XML B2B, Java & XML, The XML Files, XML & WML, Voice XML, SYS-CON Radio, XML & XSLT, XML & XSL, XML & XHTML, 2B or Not 2B, XML Industry Insider, XML Script, <e-BizML>, XML & Business, XML Demystified, XML & E-Commerce, XML Middleware, and much more!

XML-J The Complete Works
~~Reg $59.99~~
Buy Online
Only $53.99

COLDFUSION Developer's Journal

The most complete library of exclusive *CFDJ* articles!

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

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

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

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

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

machine. When the proxy is created using the WSDL utility (see Listing 5), the namespace AreaWeatherWS is specified using the switch /n:AreaWeatherWS. This will be imported later into our consuming application's code. After ProxyCompiler is run, we must copy the resulting AreaWeatherWS.dll into the bin directory mentioned earlier. Now we're ready to have a consumer make use of it.

The Mobile Page: SojournerMobilePage.aspx

The mobile page code in Listing 6 contains many of the same elements we addressed previously, such as the Page and Register directives. However, we have some new items to discuss. First, there's an Import directive that includes a Namespace attribute with the value AreaWeatherWS. This is how the mobile page makes use of the Web service.

This is followed by the frmMain mobile form, which is initially displayed when the mobile page is called. The first two labels are used for identification and instructions. The textbox is where the user enters the postal code, city/state, or city/country combination. Finally, the command button executes the ViewWeather routine, which makes a call to the Area Weather Web service for the weather information.

The second mobile form contains only two elements: a label (lblWeather) that will contain the weather information and a command button (btnBack) to take the user back to the main form by executing the ReturnToMain routine below.

The C# server-side script contains two functions. The first makes frmMain the active form so it'll be displayed when the user selects the btnBack command button. The second routine is where the bulk of the work occurs. Since the namespace AreaWeatherWS was declared above, we can instantiate it within the mobile page as the object WeatherService. (We could have called it anything we wished.) With this namespace we have the GetWeatherClass available for the application's use. To remove the unwanted HTML from the weather information string provided by the Web service, a series of regular expressions functions transform the string into a suitable format. This string is inserted into the Text property of the label control (lblWeather.Text). Finally, the frmWeatherData is made the active form and the weather information is populated on the screen.

Now that we've verified that the application works the way we want it to, we can use a different mobile device emulator to test it (the Nokia 6210, available from YOURWAP.com). Let's take a look at its interface in Figure 5. We'll use the postal code from the previous example, 16001, for Butler, Pennsylvania. We can see the succession as we go from left to right, top to button. Once the Go! button is pressed, weather information will be returned. We can try this again by pressing the Back button and choosing "Paris, France", "90210", "Topeka, Kansas". Please note that all locations may not be available.

Conclusion

The Sojourner mobile application doesn't have the rich display of standard Web sites. The instructions are terse, and there's little to provide the optimal user experience. However, with mobile applications we must keep in mind that screen real estate and speed are two issues that must be placed at the top of the priority list when developing for mobile devices. This is addressed quite well, minimizing the amount of user entries required and displaying information in an easy-to-read format. This allows our users to travel as they must and have access to the information they need.

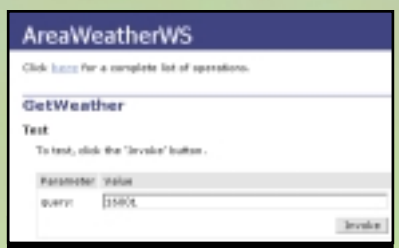


FIGURE 3 WSDL test page


FIGURE 4 Web service test results

To improve on this solution, the next step would be to have the device intermittently receive current coordinates from a GPSS satellite based on the user's current position. As it receives coordinates, the device would then send these values to the Web service as movement occurs, and information would be displayed on the device in a real-time fashion, thus requiring minimal user intervention.


FIGURE 5 Nokia 6210 emulator

• • •

I trust this has been a useful introduction to ASP.NET mobile controls and, in particular, how a Web service can be used to create a viable mobile solution. By serving as a data retrieval component, the Web service frees the mobile page to handle the display-related issues. In this way we conform the application to an appropriate separation of presentation and business logic.

Additional Information

More information about ASP.NET mobile controls and the .NET Framework is available in *ASP.NET Mobile Controls: Adaptive Web Content for Mobile Devices with the MMIT* (Wrox).

You can download the Microsoft Mobile Internet Toolkit for use with the .NET Framework from http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/MSDN-FILES/027/001/516/msdncompositedoc.xml.

To download the Microsoft Mobile Emulator, go to www.microsoft.com/MOBILE/phones/mme/mmemulator.asp.

Download the YOURWAP suite of emulators from www.yourwap.com.

Sources

- ***The Weather Underground:*** www.wunderground.com
- Sussman, D., et al. (2001). *Professional ASP.NET*. Wrox.

Author Bio

Neil Raybould is a software developer with Crossoft Inc., which specializes in intranet development. Vice president of the Pittsburgh .NET Developers' Group (www.pghdotnet.org), Neil coauthored ASP.NET Mobile Controls: Adaptive Web Content for Mobile Devices with the MMIT, Beginning ASP.NET Using C#, *and* Beginning ASP.NET Using VB.NET *(all published by Wrox).*

NRAYBOULD@CROSSOFT.COM

"With mobile applications screen real estate and speed must be placed at the top of the priority list"

LISTING 1 Compiler directives

```
<%@ Page language="C#" Inherits=
  "System.Web.UI.MobileControls.MobilePage"%>
<%@ Register TagPrefix="mobile" Namespace=
  "System.Web.UI.MobileControls" Assembly=
  "System.Web.Mobile" %>
```

LISTING 2 Server-side controls

```
<mobile:form id="Form1" runat="server">
 <mobile:TextBox id="TextBox1" runat="server"
       Text="This is Form1." />
 <mobile:Command id="Button1"
       onclick="ButtonRoutine"
       runat="server" text="Go!"></mobile:Command>
</mobile:form>
<mobile:form id="Form2" runat="server">
 <mobile:TextBox id="TextBox2" runat="server"
       Text="This is Form2." />
</mobile:form>
```

LISTING 3 Server-side script

```
<script runat="server">
void ButtonRoutine(object system,
                   System.EventArgs e) {
 ActiveForm = Form2;
}
</script>
```

LISTING 4 AreaWeatherWS.asmx

```
<%@ WebService Language="VB"
    Class="AreaWeatherWS" %>
Imports System
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Xml.Serialization

'This class makes a request to wunderground.com
'and retrieves weather information based on a
'postal code or area name
<WebService(Description:="Sojourner Web Service",_
    Namespace:="http://localhost/Sojourner/")>_
Public Class AreaWeatherWS
  Inherits System.Web.Services.Protocols.
           HttpGetClientProtocol
  Public Sub New()
      MyBase.New
      Me.Url = "http://www.wunderground.com/cgi-bin/
               findweather/getForecast"
  End Sub
  <WebMethod(),_
   HttpMethodAttribute(GetType(TextReturnReader),
   GetType(UrlParameterWriter))> _
  Public Function GetWeather(ByVal query As String)
                  As GetWeatherMatch
        Return CType(Me.Invoke("GetWeather", _
               (Me.Url), New Object() {query}), _
               GetWeatherMatch)
  End Function
End Class

'This class uses regular expressions to find
 specific text within the HTML source and
 will return it as a string
Public Class GetWeatherMatch
  <MatchAttribute(
   "<tr ><td>Temperature</td>\n<td><b>(.*)
   <tr ><td>Moon Phase</td><td>",
   IgnoreCase:=true)> _
   Public Weather As String
End Class
```

LISTING 5 ProxyCompiler.bat

```
path=c:\program files\Microsoft.NET\FrameworkSDK\bin;
c:\winnt\Microsoft.NET\Framework\v1.0.2914

wsdl.exe
      http://localhost/Sojourner/AreaWeatherWS.asmx?wsdl
      /l:vb /n:AreaWeatherWS
vbc /out:/bin/AreaWeatherWS.dll /t:library /r:System.dll
      /r:System.Xml.dll /r:System.Web.Services.dll
      AreaWeatherWS.vb
```

LISTING 6 SojournerMobilePage.aspx

```
<%@ Register TagPrefix="mobile" Namespace=
  "System.Web.UI.MobileControls" Assembly=
  "System.Web.Mobile" %>
<%@ Page language="C#" Inherits=
  "System.Web.UI.MobileControls.MobilePage" %>
<%@ Import Namespace="AreaWeatherWS"%>
<mobile:form id="frmMain" runat="server">
 <mobile:Label id="lblMainCaption" runat="server" Text=
  "SOJOURNER" /><br>
 <mobile:Label id="lblPrompt" runat="server"
       Text="Please enter postal code
             or country name" />
      <br>
 <mobile:TextBox id="txtEntry" runat="server" />
      <br>
 <mobile:Command id="btnGo" onclick="ViewWeather"
       runat="server" text="Go!" />
</mobile:form>
<mobile:form id="frmWeatherData" runat="server">
 <mobile:Label id="lblWeather" runat="server" />
 <mobile:Command id="btnBack"
       onclick="ReturnToMain"
       runat="server" text="Back" />
</mobile:form>
<script runat="server">
void ReturnToMain(object sender, System.EventArgs e)
{
 ActiveForm = frmMain;
}
void ViewWeather(object sender, System.EventArgs e)
{
 AreaWeatherWS WeatherWebService =
          new AreaWeatherWS();
 GetWeatherMatch match;
 match =
          WeatherWebService.GetWeather(txtEntry.Text);
 string WeatherResults = match.Weather;
 WeatherResults =
          Regex.Replace(WeatherResults,"&#176;","");
 WeatherResults =
          Regex.Replace(WeatherResults,"<td>","");
 WeatherResults =
          Regex.Replace(WeatherResults,
                        "</td></tr>","...");
 WeatherResults =
          Regex.Replace(WeatherResults,"<tr >","");
 WeatherResults =
          Regex.Replace(WeatherResults,"</td>","");
 WeatherResults =
          Regex.Replace(WeatherResults,"<b>","");
 WeatherResults =
          Regex.Replace(WeatherResults,"<BR>","");
 WeatherResults =
          Regex.Replace(WeatherResults,"<br>","");
 WeatherResults =
          Regex.Replace(WeatherResults,"</B>","");
 WeatherResults =
          Regex.Replace(WeatherResults,"&lt;","<");
 WeatherResults =
          Regex.Replace(WeatherResults,"&gt;",">");
 WeatherResults = "Temperature " + Regex.Replace
  (WeatherResults,"</b>","");
 lblWeather.Text = WeatherResults;
 ActiveForm = frmWeatherData;
}
</script>
```

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

Parsing Filters

Web applications often need their HTML/XML documents cleaned asynchronously or synchronously. "Cleaning" could be checking for well-formedness, manipulating an expression in the documents, or parsing them for custom-specific functionality. To my knowledge, there are no specific tools that can be molded to cater to custom-specific functionalities in terms of flushing documents in their own custom way. So why not write your own parsing tool – one that could be specific to your application logic and also have the capabilities of seamlessly integrating and/or wrapping with other existing tools?

Written by Ravi Akireddy

This article presents a simple and open framework that eases custom parsing and checking of any type of document – XML, HTML, text, and so on – through the concept of filters and XML-based configurations. The idea here is to define filters, tie them up in a piped fashion, and stream the input documents through these chained filters – and in the process of streaming, make each filter apply its specific logic on the same streamed source at the same time.

A filter is meant for a specific function, and each one serves its own purpose of massaging the provided input stream. A filter could be an implementation of your own specific logic or a wrapper around an existing tool. If you're familiar with the Unix operating system, the concept appears similar to its piping feature. The difference could be that in this case the parameters are supplied through an XML-based configuration file.

Filters are made Runnable instances and run as threads. This helps in the simultaneous execution of all the filters in parallel in the input stream. Filters are aggregated in a chain and are applied simultaneously, similar to the concept of pipes in Unix.

Piping Diagram

Now let's jump into the details of a filtering framework. The piping shown in Figure 1 is a rough version of what we're going to look into. The filters to be applied are identified through a configurable XML document and are connected by piping the input and output of each filter. Once the pipe is established, input documents are streamed through these piped filters to the desired location. As the documents are being streamed, each filter applies its own filtering logic on it in parallel. The filtering framework provides plug-and-filter functionality by simply manipulating the configuration file. You can apply some filters for one class of documents and other filters for other classes of documents by modifying the configurations. If you need a new filtering feature, write one and configure its details in the configuration document – the rest is transparent.

Design

The class diagram for the filtering parser is shown in Figure 2. It has a filter class that serves as an interface for all the custom filters to implement. The filter interface has methods for setting input stream, output stream, a filtering message, and the filtering logic function. Thus each filter reads from its input source, applies its specific filtering logic, and writes it into its output stream.

If you want to add your own custom filter, you have to implement this interface and modify the configuration file for the filter to apply its logic. The AbstractFilter class implements Filter and takes care of the common functionality implementations for the specific filters to inherit. It implements the functionality of setting the input stream, the output stream, and the filter message.

FilterManager is responsible for parsing the configured filters, loading the filter classes, and stitching them together. It uses an inner class FilterLoader for parsing the configurable filters file. The remaining classes are the specific filters implemented as part of this framework and are included in Listings 1–8, which are downloadable at www.sys-con.com/xml/source.cfm. You can see that each filter class is meant for a specific purpose. That's the entire framework…isn't it simple? Let's see how powerful it really is.

Figure 3 is the sequence diagram for the filtering parser.

FilterManager takes the following inputs to start the filtering process:

- The *input* directory location from which the documents are to be read for processing
- The *output* directory location into which the filtered documents have to be written

When instantiated, FilterManager reads a configuration file, instantiates all the required filters, and chains one to the other. The configuration file filters.xml contains the details of the parsing filters that are to be applied to the input files. When FilterManager is invoked to run the filters, it does the following:

- Loads the configured filters
- Sets the input and output stream on each filter
- Pipes the filters by connecting the output of one filter to the input of the next one
- Starts each filter
- Reads the documents from the source location and streams them through the filter's pipe to the destination location

Let's look into the details of the implementation.

Configuring Filters

Filters are configured through an XML document, filters.xml. If a filter has to be applied, its details have to be added to this configuration file. The document root, filters element, contains one or more filter elements. A sample filter fragment is shown in Listing 1.

Each filter element should have the following attributes at a minimum:

- ***Name:*** Name of the filter – used for convenience
- ***Filterclass:*** Fully qualified class name of the filter that will be loaded and used while parsing

Configuring these filters is so flexible that, except for the attributes name and filter class, there's no specific DTD/Schema on the filter element for a filter to stick to. Also, there's no restriction on the content that a filter element should encapsulate. The idea here is to hand over the filter nodes to the respective filter class – it's up to the filter to analyze the node. The fragment details in a filter element thus make sense only in the context of that filter. This way you get the flexibility of defining your own filter fragment and implementing a filter class that transforms the input stream based on the filter content that it could understand. This puts everything in your control and you get plug-and-filter functionality the way you want it.

If you look at the filter fragment details shown in Listing 2, the filter name is "FindAndReplace" and the filter class to be used is "filters.FindAndReplace". The encapsulated content – the child elements *find* and *replace* of the filter "FindAndReplace" – will be understood only by the class filters FindAndReplace. This defines the filter-specific parsing message that will be used by that particular filter while parsing the input streams.

Filter classes are loaded while parsing the filters.xml file, and each instance of the filter is set with its own filterable parameters. These filterable parameters are nothing but the content part of that particular filter fragment. Setting the filter node as a

message on that filter class does this. As there's no restriction on the content of the filter node, you can put any amount of data in each one. The individual filter encapsulates the implementation details of unmarshaling the entire content within the corresponding filter element. Thus a filter analyzes this data – the message node – and sets its filterable parameters.

The document filters.xml contains the details of the filters to be applied to the input stream. There's no particular sequence of filters to follow. Users can add/delete/update the filter details as they wish, but should encapsulate certain details of the filter, that is, a particular filter should be in a position to understand the filterable details while unmarshaling the message.

Filters

Filter is an interface for all custom filtering classes to implement. This extends the Runnable interface so each filter can be executed in its own thread. Filters are made to work as threads so we can pipe the output stream of one filter into the input stream of another and make them work in parallel. This way all the filters work at the same time, in parallel, on each input stream and finally dump the resultant output into the desired destination. Filter class method declarations are shown in Listing 3.

Filters expose functionality for setting input and output streams and also a message to set on them. As discussed previously, a message is nothing but the filter fragment of the configuration file for that filter represented as a node object.

The AbstractFilter class is a convenient class that implements the common functionality for specific filters to inherit. Here, it implements the functionality for setting the input and output streams as well as the message node object and the run() method of java.lang.Runnable.

Implementing the method run(), shown in Listing 4, of the AbstractFilter class is simple: it calls the doFilter() method. Function doFilter() is an abstract function that has to be implemented by the custom filter(s). The crux of filtering is thus the implementation details of this function. All the specific filter classes are required to implement this method to carry the specific logic of filtering.

The implementation details for this function are to:

- Unmarshal the message node if it has one and set its filterable parameters.
- Apply the specific logic on the input stream based on the unmarshaled parameters and write it into the output.

The implementation of AbstractFilter is shown in Listing 4.

FilterManager

The FilterManager class is the one that does the dirty work of extracting the filter details, stitching the filters, reading the input sources, and executing the filters. It has an inner class, FilterLoader (see Listing 5), that loads the configuration file (filters.xml), parses the file, identifies the filter classes, loads them, and aggregates them in a vector. Once loaded, FilterManager uses this vector of filters to create the filters pipe and stream the source documents.

Method getFilterDocument() of class FilterLoader (see Listing 5) uses the DOM API to parse the filters.xml file and returns its document object.

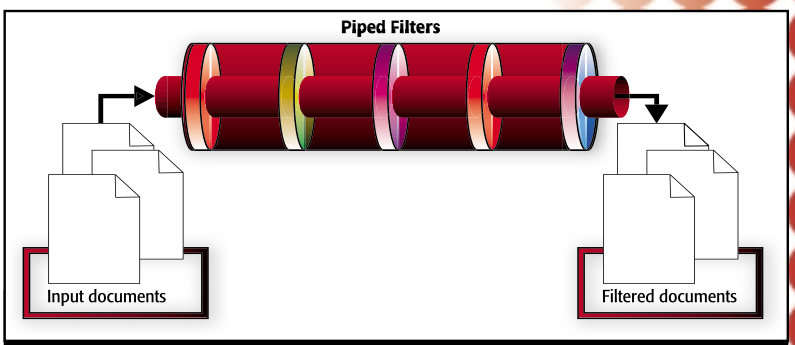


FIGURE 1 Filter piping diagram

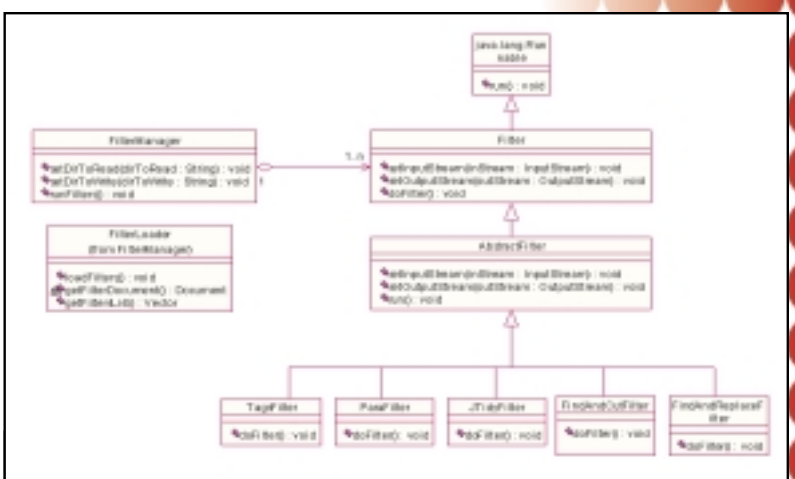


FIGURE 2 Class diagram

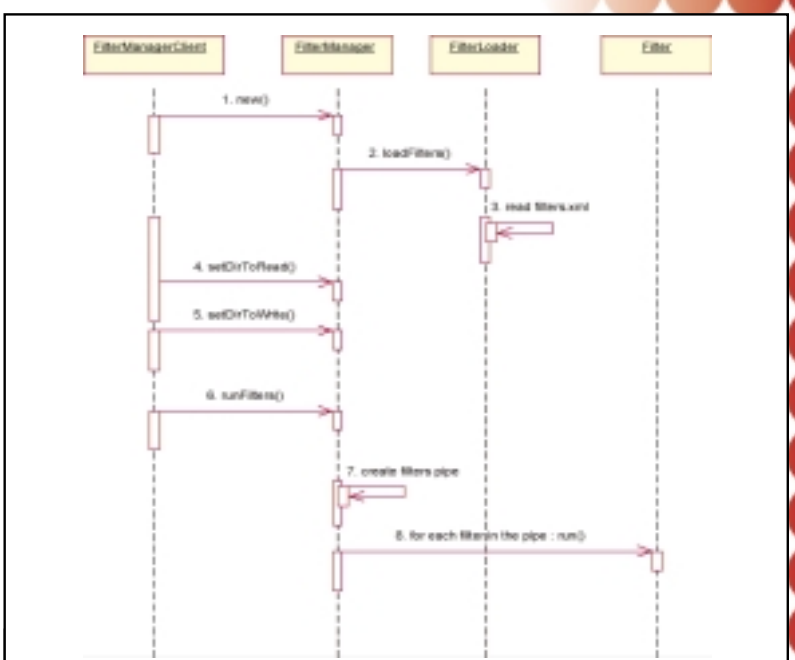


FIGURE 3 Sequence diagram

AUTHOR BIO

Ravi Akireddy, a senior software engineer at Comergent Technologies, Inc., has worked extensively on CORBA, Java, and XML-related technologies. Author of a number of articles on XML and Java programming, Ravi's interests include object-oriented analysis and design, software development methodologies, and Jini.

The loadFilters() method gets the document object, loops through all the filter fragments, loads the individual filter classes, and creates the respective filter objects. The filter objects are appended to a vector, filterTable, as they're loaded.

Once a filter is loaded, the corresponding filter fragment of filters.xml is set as a message node on that filter. This message node object will be unmarshaled by the individual filter before filtering the input stream and used for its filtering process. For now, message node is nothing but the entire contents of that particular filter element as a DOM node object. We'll see in the section "Sample Filter" what it means to a specific filter instance.

Each filter takes an input stream and outputs a stream after applying the filter-specific logic. Filtering parameters for each filter is set as part of initializing the filter chain by parsing the configurable XML file. The message is a document node that encapsulates filterable parameters and makes sense in the context of that filter. Each filter unmarshals the message node and sets its filterable parameters. Element filter in the filters document filters.xml serves as the message node for that particular filter. While looping through the filters (filters.xml) document object, we extract the individual filter class and set the same filter context on that filter object as message node.

If you look at the details of the FindAndReplace element (see Listing 1), it has attributes, a name whose value is FindAndReplace, and a filter class with filters. FindAndReplaceFilter, which carries the implementation details of finding a given pattern and replacing it with another pattern. The FindAndReplace filter has two child elements,"find" and "replace." While unmarshaling, only FindAndReplaceFilter can understand what these two parameters are meant for.

Similarly, if you take FindAndCut filter, it has name and filter class attributes, but only one child element, called "findandcut." The filter class filters.FindAndCut has the specific details to analyze the message node and set its filterable parameters. The content of each filter element thus makes sense only in the context of that filter. Hence it is highly customizable and can easily be extended.

Types of Filters

To show the ease of filter implementation, I've written some quick filters that are included in the listings, with a brief description of each.

- ***FindAndReplaceFilter:*** Intercepts the input stream and sniffs for the "find" parameter and replaces with the "replace" parameter once found
- ***FindAndCutFilter:*** Trims a set parameter on the input stream and writes into the output stream
- ***ParaFilter:*** Specific to HTML-specific input streams – makes the input documents well formed with respect to para <p> tags
- ***CaseChangeFilter:*** Modifies the case of a given parameter string in the input stream; the output stream will be the input stream but with a case change for that particular parameter
- ***TagsFilter:*** Applies the configured tags for the matched expression in the input stream; example: look for string "I am bold" and write the output as "I am bold"
- ***JtidyFilter:*** A wrapping filter on Jtidy tool – illustrates the ease of integrating external tools seamlessly into the filters framework

Running the Filters

To run the FilterManager, you have to supply both an input and an output directory. The former is where the FilterManager reads the files; the latter is where the filtered files are written. Listing 6 from the FilterManager class has the implementation details for setting the directories as well as for executing the piping sequence of the filters.

The method runFilters() contains the logic for reading the input files one by one and streaming them through the filters pipe. It has an anonymous class that implements the accept method of java.io.FilenameFilter class, which returns an array of interested files in the input directory. Listing 6 contains the implementation details of filtering HTML files only. However, you can modify the if statement to aggregate more than one type of document.

Having gotten the list of files, we have to apply the filters configured in the configuration file filters.xml . To do that, I've created a streaming pipe by attaching filters to each other. As each filter takes an input and output stream, it's easy to pipe them through the powerful Java streaming API, namely java.io.PipedOutputStream and java.io.PipedInputStream classes. The OutputStream of a filter is piped to the InputStream of the next filter, as follows:

```
pipeOut = new PipedOutputStream();
pipeIn = new PipedInputStream(pipeOut);
out = new PrintStream(pipeOut);
next.setInputStream(in);
next.setOutputStream(out);
```

where the variable "next" is a filter object and "in" is the pipeIn of the previous filter object.

This way you loop through all the filters and expand the filtering pipe. The InputStream for the first filter will be the file stream from the input directory. As each filter is a Runnable instance, they're started as threads while constructing the pipe. So now the filters are started and piped. They're ready for reading the input stream and apply their respective filtering logic in parallel.

We start reading the stream through the final filter object, as follows:

```
int nextChar ;
while ((nextChar = finalReader.read()) != -1) {
  finalWriter.write(nextChar);
}
```

where the variable "finalReader" is the reader on the final filter object and the variable "finalWriter" is the writer to the output destination file.

A "read" operation on the finalReader is a read on the entire piped filters; in the process of reading, each filter is applying its filtering logic. Finally, it's written into the corresponding output file. This is done in a loop for the entire list of files obtained from the source directory (files directory). The filtered files are written into the provided destination directory with file names that are the same as the original file names. In the source provided, it is hard-coded to write into the "out" directory under the "files" directory.

Running FilterManager

The FilterManager class has an inner static class, FilterManagerClient, that instantiates and runs the FilterManager. The downloadable code has a batch file, runfilters.bat, that runs the FilterManagerClient. The following is the FilterManagerClient class:

```
public static class FilterManagerClient {
public static void main(String []args) throws Exception {
    FilterManager fm = new FilterManager();
    fm.runFilters();
}
}
```

Sample Filter

Now that we've discussed filters and how they're piped, let's look into the details of a sample filter. Listing 7 gives the details of FindAndReplaceFilter, which finds a string and replaces it with another string. The parameters are set while reading the filters.xml.

The abstract method doFilter() is the one that has to be implemented; the remainder of the setters and getters are inherited from the AbstractFilter class. Once doFilter() is invoked, it first sets the filterable parameters, that is, the find and replace variables, by extracting the elements find and replace from the node object, which is set while instantiating the filter. Here the content of the node makes sense only with respect to this filter and gives us the flexibility of writing our own filters and implementing the specific logic.

A read on the filtering pipe is a read here. The logic in this filter is to read a line and pass it to method findAndReplace(String), which scans for the find parameter and replaces it with the replace parameter. Once the entire line has been scanned, it returns the filtered line and is written into the output stream, which is nothing but the piped input stream for the next filter.

Once this filter has applied its logic, the stream is fed to the next filter. Meanwhile, this filter is busy reading the next line from the previous filter. This way all the filters are working simultaneously on the same stream.

How Simple Is It?

Filters can be your own implementation of the filter class or a wrapper on any existing implementations or tools. Let's look into both categories – custom filters and wrapping filters.

Custom Filters

A new filter should contain the implementation details for both the doFilter() and parseMessage() methods.

doFilter() will contain the core logic of reading the input stream, applying its filtering logic, and, once that's done, writing into an output stream.

parseMessage() should implement the logic of parsing and analyzing the filter node details and set its internal variables, which may be used while applying the filtering logic in the doFilter() method.

External Filters

Fine, you say. This works great when you write your own filters. But how do you integrate with a third-party tool and make your filters work seamlessly? Here's an example that shows how simple and easy it is to integrate an available tool.

I took Jtidy, an excellent tool that converts the documents into XHTML format. I wrote the JtidyFilter class, which is nothing but a wrapper on the Jtidy main class. I put the necessary code in the doFilter() function…and that's it. It works just like any other filter. It takes an input stream, cleans it, and writes to the output stream, which is fed to the next filter.

To make this filter participate in the filters piping, you have to add the filter details to the configuration file filters.xml file. The implementation details for the doFilter() method of the JtidyFilter class are shown in Listing 8.

The following XML fragment shows the details you have to add to include JtidyFilter in the filtering sequence:

```
<filter name="jtidy" class="filters.JTidyFilter"/>
```

It has the name of the filter and the filter class to use and no child fragments. So the JtidyFilter class doesn't have to analyze the message content. But you can always provide some initialization parameters for the underlying tool through the content of the respective filter fragment. The rest is taken care of by the FilterManager class. Isn't it simple?

Transformation Filter

To show you how easy it is to plug a transformer/SAX2 XMLFilter into this framework, and how seamless it would be, I took the sample SAX2 XMLFilter provided in the Xalan downloads and copied it into the TransformFilter.java class. After renaming the main() function to doFilter(), I tweaked it a little to adjust the input and output and added a new function, parseMessage(), which sets the stylesheet(s) to use for transforming.

I added the following XML fragment to the filters.xml:

```
<filter name="transformer" class="filters.TransformFilter">
    <stylesheet>echo.xsl</stylesheet>
</filter>
```

The content of the filter encapsulates the stylesheet(s) to use to transform the input. That's all I did. In less than half an hour I was able to plug in the transformation filter and test it.

Conclusion

I've tried to simplify the logic of parsing any kind of document in a custom-specific fashion. This way you can have a mix and match of filtering logic for a variety of techniques. You can apply the logic for making well-formed documents or regular expression filtering or any other kind of massaging to documents. The guts of each one lie in the specific filter and each works independently. To apply a new logic, write a filter and update the filter's configuration XML document; the rest is transparent – it's as simple as that. So, when there are no parsers, write your own.

Enjoy filtering…

RAKIREDDY@COMERGENT.COM

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 INTEGRATION

Sponsored by:

IONA | END 2 ANYWHERE™ · TogetherSoft · ADDS · bea · SilverStream · MERANT · Federal Computer Week · SD Times · Charles F. Goldfarb's XML TIMES.com · jCert · wrox · Webservices.org · Java Skyline · WebServices JOURNAL · JAVA DEVELOPER'S JOURNAL · XML JOURNAL · wireless BUSINESS&TECHNOLOGY · BEA WebLogic DEVELOPER'S JOURNAL · ColdFusion Developer's Journal · WebSphere DEVELOPER'S JOURNAL · CF Advisor

Sponsored by: SYS-CON MEDIA

Produced by: SYS-CON EVENTS

Web Services – Skills, Strategy, and Vision

For the developer. the latest tools and techniques...

For the architect, the latest designs....

For the VP/CIO, technology management issues

- Invaluable information in the form of discussions, presentations, tutorials, and case studies
- Unmatched Keynotes and Faculty - gurus in the Java, XML, .NET, and Web Services world
- The largest independent Web Services, Java, and XML Expos
- An unparalleled opportunity to network with over 5,000 *i*-technology professionals

New in 2002!

Each track will feature "Hot Breaking" Sessions to keep attendees up-to-the-minute on Emerging Technologies and Strategies!

Featuring 2 Keynote Panels and Industry Perspectives from the Visionaries shaping Next Generation Technologies and Business Strategies

2002

ONLINE
EARLY BIRD REGISTRATION NOW OPEN!
SAVE $400

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

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

FOR MORE INFORMATION
SYS-CON EVENTS, INC
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645

201-802-3069
WWW.SYS-CON.COM

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

Who Should Exhibit…
Java, XML, Web Services, and .NET Technology vendors, staking their claim to this fast-evolving marketplace

Web Services Edge Conference Committee

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

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

Ajit Sagar
XML Track Chair
Editor-in-Chief
XML-Journal

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

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

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

Andy Astor
IT Strategy Track Chair
International Advisory Board
Web Services Journal

IT Strategy Track
- Key Architectural Issues in a World of Web Services
- .NET vs J2EE – From Religious Wars to Fact-Based Decision Making
- Getting Started with Web Services
- Selecting a Framework: Toolkit, Platform, or Roll Your Own?
- Infrastructure Vendors–A Map of the World
- Extreme Programming and Web Services Projects: Defining ROI
- Standards You Need to Know About
- Best Practices You Need to Insist On
- Introduction to Business Rules and Rules Engines
- Panel Discussion- Web Service Business/ Economic Models

Jim Milbery
Vendor Track Chair
Product Review Editor
Java Developer's Journal

Vendor Track
- .NET, J2EE, JMS and other Messaging
- Case and Development Tools
- The Use of Testing Tools
- How-to Demonstrations

[WRITTEN BY HITESH SETH

An overview

Tools for Developing VoiceXML Applications

Part 2

Last year in this column (***XML-J***, Vol. 2, issue 2) we reviewed an initial set of VoiceXML development tools and integrated development environments (IDEs). Now we'll review some of the new tool sets developed by independent software vendors for the development and testing of dynamic speech-based components and applications.

I'll focus on how to use these tools to test and debug VoiceXML applications from a desktop-based development environment and/or a normal touch-tone–based phone.

Note/Disclaimer: This article doesn't attempt to compare the capabilities and features of the various tools, but instead provides information on the various development tools in a concise fashion. To ensure neutrality, the vendors have also been listed in alphabetical order. Even though I've tried to provide the latest and most correct information and compliance, please validate it with respect to the VoiceXML specifications through the appropriate vendors.

GetVocal VoiceXML SDK

VoiceXML SDK from GetVocal (a division of Digital Voice Technologies) is a desktop-based Visual VoiceXML development and testing tool. It supports the latest (VoiceXML 2.0 draft) version of the specification including support for XML-based grammars. It's built on top of Microsoft Speech SDK 5.1 and uses Microsoft TTS and ASR engines for text-to-speech and speech recognition.

Graphical testing and development is the key highlight of the SDK (see Figure 1), which also includes a virtual phone pad for real-phone simulation, a graphical VoiceXML generator, scripting support through JavaScript VBScript (with extensions for invoking ActiveX components and Java classes), and a complete visual debugging environment with steps, breakpoints, and variable watches.

Apart from testing local VoiceXML content, the SDK also supports visual interpretation and testing of remote/dynamic VoiceXML applications. A distinctive highlight of the SDK is that apart from the simulation through a virtual phone, it can actually use a TAPI-based interface to interact with a live phone through a voice modem or an installed dialogic card. This feature converts the SDK from a desktop-based testing tool to a full-featured single-port VoiceXML gateway. The SDK also includes a complete VoiceXML tag reference and sample VoiceXML applications.

Included with the SDK is GetVocal VoiceXML Editor, a rich editor that features syntax highlighting and configurable tag shortcuts. A time-limited evaluation version of GetVocal SDK is available for Windows 98/ME/NT/2000/XP platforms from www.getvocal.com/vxml.asp.

HeyAnita FreeSpeech Developer Network

FSDN is a hosted VoiceXML platform. The tools include integrated development environments for developing VoiceXML applications and XML-based grammars. The Web-based integrated development environment approach provides flexibility by allowing VoiceXML documents and external grammars to be stored in multiple locations: a self-managed online scratchpad, uploaded files, and external Web servers.

Applications built using the FreeSpeech Developer Network use XML

Update: VoiceXML 2.0 Draft Specification Is Now Available

In October 2001, W3C released a draft of VoiceXML 2.0. VoiceXML 1.0 was released by the VoiceXML Forum in March 2000. Some highlights of the 2.0 draft:

- The W3C Voice Browser Activity (www.w3.org/voice) has officially taken over the management and technical development of the VoiceXML specification. This is a good step for the development of VoiceXML as W3C is a great standards organization. The VoiceXML Forum (www.voicexml.org) continues to focus on conformance testing and a branding program for VoiceXML.
- Mandatory support for XML-based Speech Synthesis Markup Language (www.w3.org/TR/speech-synthesis).
- Mandatory support for XML-based Speech Recognition Grammar Format (www.w3.org/TR/speech-grammar). This is critical as it will result in better conformance between the different vendor implementations of VoiceXML 2.0. In addition to the XML-based format, VoiceXML 2.0 implementations can optionally support the augmented BNF form of the grammar format specification as well as other formats such as JSGF.
- A number of other changes have been recommended, such as changes in both elements and attributes.

AUTHOR BIO

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

grammars that are based on the Speech Recognition Grammar Specification draft available from W3C (www.w3.org/TR/speech-grammar), now part of VoiceXML 2.0 (see sidebar). To jump-start grammar development there are a couple of tools such as a wizard-based grammar generator (see Figure 2) and an online grammar parser to test utterances.

To assist in development, FSDN provides technical resources such as a VoiceXML reference, a grammar reference, FAQ, tutorials, and a prebuilt VoiceXML and grammar library. For communication between developers, FSDN includes NNTP-based threaded forums. FreeSpeech Developer Network is available at http://freespeech.heyanita.com.

IBM WebSphere Voice Toolkit

WebSphere Voice Toolkit is a desktop-based IDE that's built on top of IBM's next-generation open-source development tool platform, Eclipse (www.eclipse.org). Voice Toolkit is really written as a plug-in to the underlying core Eclipse platform. This allows Voice Toolkit to inherit the advanced IDE features provided by the open-source Eclipse IDE and IBM's commercial tools built on top of Eclipse. For instance, Voice Toolkit can inherit other Eclipse modules (plug-ins) for server-based dynamic application development with IBM WebSphere Application Developer Studio to provide an integrated server-based J2EE/VoiceXML application development with full-fledged features for J2EE application development and common IDE features such as CVS version management and project management.

For VoiceXML applications, Voice Toolkit leverages IBM Voice SDK (reviewed in this column in *XML-J*, Vol. 2, issue 2) to interface with desktop text-to-speech and recognition engines. This integration enables the toolkit to test the VoiceXML content in both audio and text simulation modes. In addition, Toolkit includes tools for editing and testing Java Speech Grammar Format (JSGF) and Speech Recognition Control Language (SRCL) based grammars (see Figure 3), a color-coded VoiceXML editor with a content assist feature to select elements and attributes, an audio recorder for creating and playing .au/.wav prompts, a pronunciation builder for the WebSphere VoiceXML Server/SDK, wizards for creating database-based dynamic Web applications using JSP and JavaBeans, and a voice application debugger.

WebSphere Voice Toolkit is available for Windows 2000–based operating systems from www-3.ibm.com/software/speech/enterprise/vtoolkit.html.

Telera DeVXchange AppBuilder

DeVXchange AppBuilder from Telera Inc. is a desktop-based visual VoiceXML application generation IDE. At first it gives the impression that it's a traditional IVR toolset, and even its drag-and-drop palette-building blocks don't directly map to VoiceXML elements. However, with a little bit of exposure you can understand that the tool focuses on generating the underlying dynamic scripting code in VoiceXML 1.0.

Figure 4 shows a simple menu-based dialog built using the visual tool. The bottom window shows the VoiceXML dynamically generated. In addition, the tool features integration with Visual SourceSafe for configuration management, FTP-based remote publishing, rich building blocks for a call-flow scheduler, a task balancer, logging, speech recording, variable-based call routing, database query, call transfer, call bridging, and the ability to create custom reusable blocks based on XML. The tool differs from a number of other VoiceXML IDEs as it focuses on the server-based ASP/JSP-based dynamic application generation rather than the VoiceXML template itself.

Telera AppBuilder runs on top of Windows NT/2000 platforms and requires either Microsoft Internet Information Server 4.0/5.0 and/or a JSP/servlet engine such as Apache Tomcat (included with the installation). More information about AppBuilder is available at www.telera.com/appbuilder.html.

VoiceGenie Genie IDE

Genie IDE from VoiceGenie Technologies is a desktop-based toolkit for developing and testing local and remote VoiceXML-based applications. Genie 1.0 (see Figure 5) is a full-featured environment for the interactive development of VoiceXML applications, including project management with FTP-based publishing support, syntax highlighting, indentation, a palette-like tag manipulation tool for easy tag reuse, configurable DTD support, and coding assistance through tag and attribute choices. The IDE also includes online help, sample VoiceXML files, and a step-by-step tutorial.

A distinct highlight and highly productive feature of the IDE is the wizard-based support for VoiceXML generation. Using intuitive and simple dialogs, wizards can help create both VoiceXML dialogs and forms and XML- and ABNF-based grammars. Genie 1.0 leverages technologies such as the latest VoiceGenie VoiceXML interpreter and the Microsoft SAPI interface for Text to Speech (TTS).

Apart from the development and testing of locally created .vxml files, Genie 1.0 IDE also supports the testing of remote (URL-based) VoiceXML applications. This feature can be used to test application server–based dynamic VoiceXML applications. In essence, the local IDE can be used to create the VoiceXML content template and additional dynamic scripts and tags (such as Perl/JSP/ASP/ASP.NET), then be served through a Web or application server and tested remotely. To aid the dialog flow, Genie 1.0 also provides a visual tool called "Show Dialog Flow," which constructs a simple flow diagram.

Genie IDE 1.0 is available officially for Windows 2000 (Service Pack 1). However, I tested my version on Windows XP and it was pretty stable. A 90-day evaluation copy of the Genie 1.0 IDE is available for download from http://developer.voicegenie.com/IDE.php.

Voxeo Designer

Voxeo Designer from Voxeo Corporation is a Java-based visual VoiceXML editor. It provides an integrated round-trip source and visual dialog design editor with a focus on visual representation of the dialog flow (see Figure 6). It uses a context-sensitive, palette-based toolbar for visual drag-and-drop–based VoiceXML generation. It includes online help and allows remote editing and publishing of VoiceXML content.

Designer is available for Windows platforms from http://community.voxeo.com/vd2.jsp.

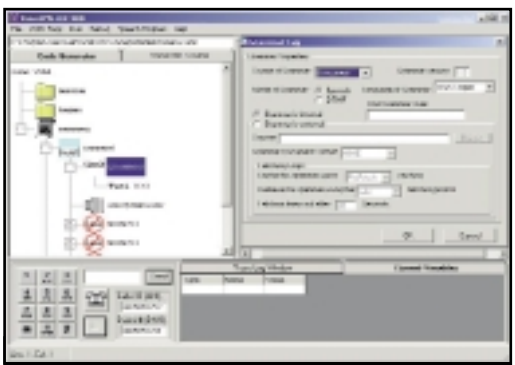


FIGURE 1 GetVocal SDK

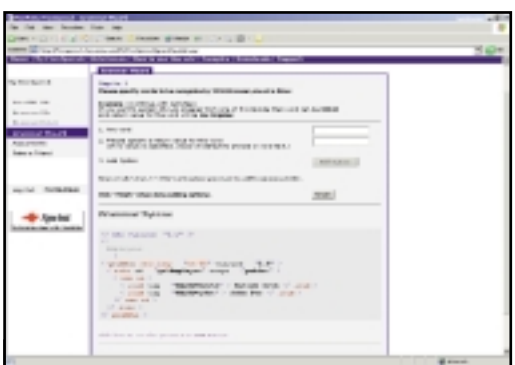


FIGURE 2 HeyAnita FSDN grammar wizard

Conclusion

The depth and breadth of these additional tool sets for interactive design, development, and testing of VoiceXML applications illustrates that the whole VoiceXML-based speech application development and deployment landscape is maturing, leveraging the advances and features that we've seen in the traditional Web application development space. This year we should see a boost in the world of VoiceXML, accelerated by the richness of some of these development tools and technologies. Table 1 provides a summary of the tools reviewed in this article.

Next I'll focus on how we can leverage Voice over IP (VoIP) to develop and deploy VoiceXML-based speech applications on IP-based telephony networks.

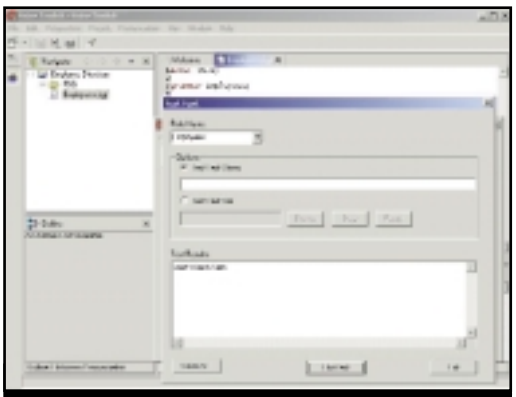


FIGURE 3 IBM WebSphere Voice Toolkit's grammar testing wizard

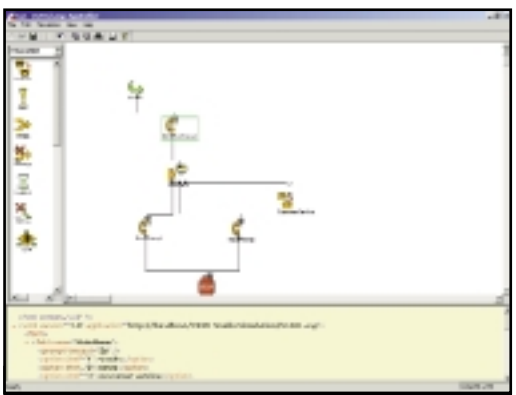


FIGURE 4 Telera DeVXchange AppBuilder

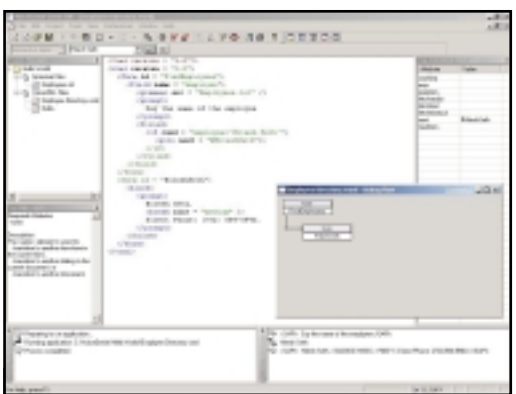


FIGURE 5 Genie IDE

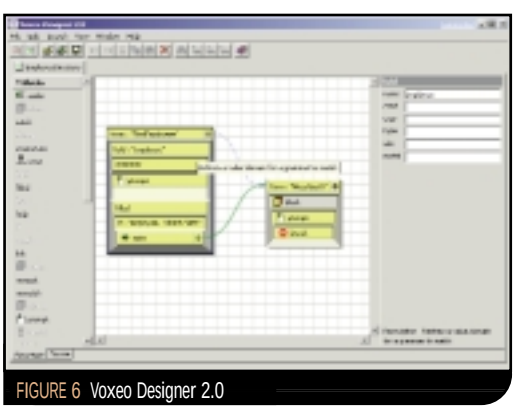


FIGURE 6 Voxeo Designer 2.0

References

- ***W3C Voice Browser Activity:*** www.w3.org/Voice/
- ***VoiceXML 2.0 Draft Specification:*** www.w3.org/TR/voicexml20/

HITESH.SETH @ SILVERLINE.COM

COMPANY	TOOL	KEY HIGHLIGHTS
GetVocal	VoiceXML SDK	• Desktop VoiceXML development IDE • Focus on visual code generation • Support for VoiceXML 2.0 features, including XML grammar format • Support for testing local and remote VoiceXML applications • Support for JavaScript and VBScript • Uses Microsoft Speech SDK 5.1 • Debugging support • Syntax-highlighted VoiceXML Editor • Functions as a single-port VoiceXML gateway using TAPI interface
HeyAnita	FreeSpeech Developer Network	• Hosted VoiceXML development platform • Support for XML-based grammar format • Support for testing both uploaded documents and remote VoiceXML applications • Wizards-based grammar generator and utterance parser
IBM	WebSphere Voice Toolkit	• Desktop VoiceXML development IDE based on IBM's open IDE platform – Eclipse • Advanced IDE features such as project management, source code control, and publishing • Support for development of dynamic VoiceXML/J2EE applications • Support for testing both local and remote VoiceXML applications • Color code VoiceXML Editor with content-assist feature • Audio recorder • JSGF/SRCL grammar editor/testing tool • Voice application debugger
Telera	DeVXchange AppBuilder	• Desktop-based dynamic VoiceXML application generation tool • Focus on visual design and generation of VoiceXML applications • Rich building blocks • Support for both ASP and JSP scripting environments • FTP-based publishing • Integration with Visual Source Safe
VoiceGenie	Genie IDE	• Desktop VoiceXML development IDE • Support for VoiceXML 2.0 features, including XML grammar format • Project management with FTP publishing support • Support for testing both local and remote VoiceXML applications • Visual dialog flow interface • Wizard-based grammar/dialog creation • Uses SAPI interface for text-to-speech • Syntax-highlighted VoiceXML Editor
Voxeo	Designer	• Java-based Desktop VoiceXML Editor • Visual dialog flow • Round-trip visual/source editing • Remote editing/publishing • Palette-based VoiceXML elements for drag-and-drop

TABLE 1 Summary of VoiceXML tools

JavaOne

http://java.sun.com/javaone

XMLSpy4.3 by Altova

[Reviewed By Jim Milbery]

Bio

Jim Milbery, a software consultant with Kuromaku Partners LLC (www.kuromaku.com), based in Easton, PA, has over 17 years of experience in application development and relational databases. He is the applications editor for **Wireless Business & Technology**, *the product review editor for* **Java Developer's Journal**, *and the author of* Making the Technical Sale.

JMILBERY @ KUROMAKU.COM

SPECS

Altova
900 Cummings Center
Suite 314T
Beverly, MA 01915
Phone: 978 927-9400
Web: www.altova.com
E-mail: sales@xmlspy.com

XML-JOURNAL XML WORLD CLASS AWARD

Test Environment
OS: Windows 2000 Service Pack 2
Processor: Toshiba Satellite Pro 866 Mhz
Memory: 256MB

The adoption rate for XML continues to be astounding. XML is finding its way into every server, into SDK, and developer platform out there, and so there is great need for developers to design XML Schemas, write XSLT stylesheets, debug Web services, and create XML content. One particular product that's getting lots of buzz is Altova's XML Spy. I recently spent an afternoon working with the latest release of this product – version 4.3.

XML Spy Suite, a product family of developer tools, consists of the XML Spy Integrated Development Environment (IDE), the XML Spy XSLT Designer, and the XML Spy Document Editor with browser plug-in. The IDE provides an intuitive, interactive interface for developers to edit and debug XML documents. The XSLT Designer is a visual tool for developing XSLT stylesheets. The Document Editor is a word-processor–like interface for creating XML content. Since the release of the first version of XML Spy in 1999, Altova has kept up with evolving XML standards, releasing product updates on average once every few months – version 4.3 is due out early this month.

Installing XML Spy Suite 4.3

You can download a trial version of the XML Spy Suite from www.xmlspy.com. Once you've downloaded the software, Altova will send you a free 30-day license key that unlocks it. XML Spy Suite 4.3 works on Windows 95, 98, NT, 2000, ME, and XP. The installation kit makes use of the InstallShield program and seems to require a reboot of your desktop during the installation process. Installation is straightforward and I was able to get XML Spy 4.3 up and running in no time.

Working with XML Spy Suite 4.3

Right off the bat I'll tell you that I'm impressed with the wide array of tools and utilities that are packaged in XML Spy Suite 4.3. Altova's developers are not shy about packing the IDE with features and functions. Yet, despite all of the functionality, the interface doesn't feel cluttered and responds briskly most of the time. (I did find that XML Spy error messages weren't always very detailed or helpful, but I didn't consider this a big issue.)

The upper left-hand section of the interface (see Figure 1) contains the project panel. You can manage multiple XML "projects" and use source control options via this project panel. The main document viewer appears in the large center panel. XML Spy provides five different viewing mechanisms:

- ***Enhanced Grid View***
- ***Schema design View***
- ***Text View***
- ***Document Editor View***
- ***Browser View***

XML Spy's Enhanced Grid View offers a powerful interface for working with XML documents. The grid splits the document viewer into two panels. The left-hand panel displays a grid outline of the open document, while the right-hand panel shows a structured version of the data within the document. If you're familiar with XML, you'll find this a comfortable WYSIWYG editing environment. Hard-core text developers will favor the raw Text View format or the Browser View format.

In the XML Spy 4 release, two new software applications were added – the XSLT Designer and Document Editor, together known as the XML Spy Document Framework. The frame-

work's goal is to provide a standards-based XML content editing and document management solution for creating and deploying large volumes of XML documents in production environments. While programmers and developers are comfortable working with raw XML data, production end users are less likely to embrace XML in this fashion. Thus Altova offers the document view within the XML Spy IDE (see Figure 2).

The Document Editor View allows end users to work with XML data within a more familiar, word-processor–type environment. Altova also offers the Document as a browser plug-in for Internet Explorer (version 5.5+). This allows end users to view and edit XML documents directly within a browser window. The Document Editor supports free-form WYSIWYG editing, form-based data input, graphics display, and real-time consistency-checking using XML Schema. Figure 2 shows a sample datasheet for a jet airplane design document that is open in the document viewer within the XML Spy IDE.

To use the Document Editor View format with your XML files you'll need to build a presentation template. Using Altova's XSLT Designer, you can build XSLT stylesheets to create the customized forms and layouts used to configure the Document Editor. XML developers who are intimately familiar with XSL elements and the XPath query language may feel comfortable using a text editor to build stylesheets. However, Altova's XSLT designer provides a GUI environment that allows the user to drag-and-drop XML elements into the design window – and the XSLT designer will automatically generate the resulting XSLT stylesheet.

The XML Spy IDE also supports database access. I was able to reverse-engineer an Access database through a Crystal Reports ODBC connection and output an XML document into my Oracle database through ODBC. I noticed that XML Spy had a tendency to vapor-lock on occasion when working with medium-size database tables – but this didn't cause me to lose any data.

Summary

I was able to find lots of entries when searching through Google's newsgroups database for mentions of XML Spy. For the most part, users seemed to like the product, and I tend to agree. Last year XML Spy won six ***XML-Journal*** Readers' Choice awards. Programmers who like a well-designed IDE will favor the XML Spy environment. On an unrelated note, I took an informal poll and all developers agree – we like the cool secret-agent cartoon girl in the Altova advertisements.

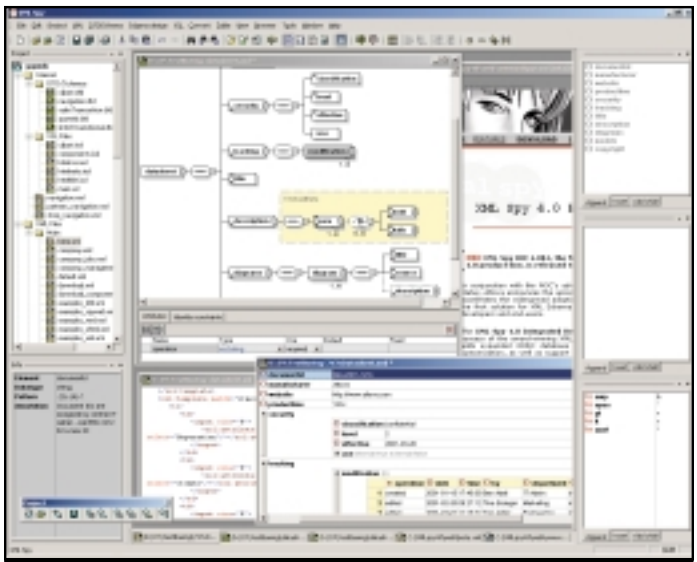


FIGURE 1 XML Spy 4 IDE

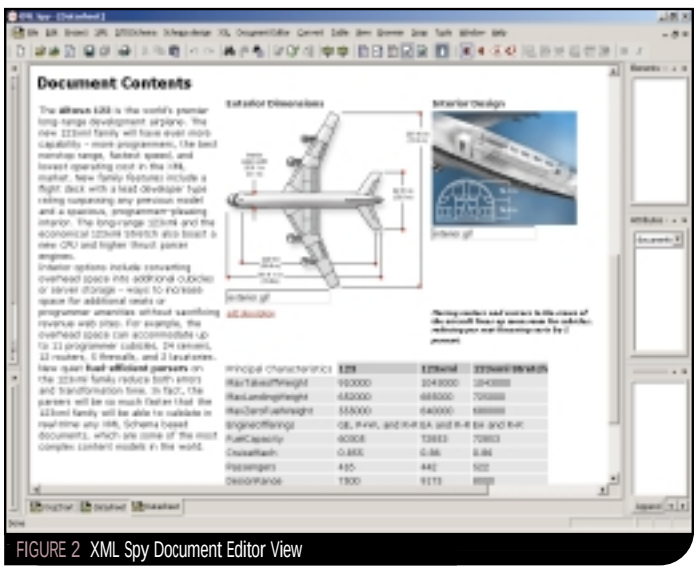
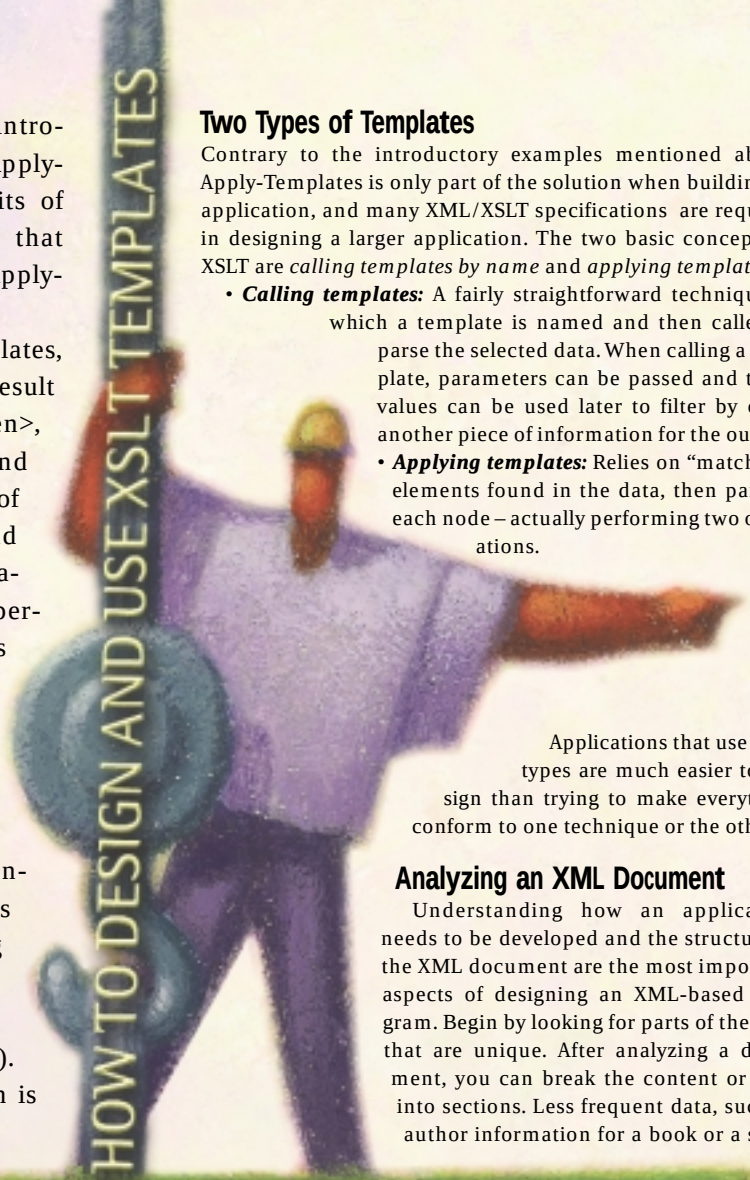


FIGURE 2 XML Spy Document Editor View

APPLY-TEMPLATES REVISITED...

HOW TO DESIGN AND USE XSLT TEMPLATES

Written by Roy Hoobler



The beginning of every XSLT book or tutorial introduces "template matching" and how to use Apply-Templates. This article explains the benefits of using XSLT templates as well as some examples that show how a "real-world" application can use Apply-Templates along with other XSLT techniques.

Previously I've skirted around using Apply-Templates, opting instead for XSLT programming logic. The result was always a lot of XSLT code with <xsl:if>, <xsl:when>, and <xsl:choose> statements. Most of the "if" and "choose" statements checked the values or existence of nodes and elements in an XML file before parsing and displaying the output as HTML or XML. These applications required a lot of logic, and most continue to perform well, so maybe not using Apply-Templates was feasible, but in the future I intend to be more careful in designing applications that use stylesheets and how those stylesheets will be developed.

I recently ran into a situation where Apply-Templates should be used – otherwise the XSL stylesheet would expand to over a hundred lines of unmanageable code. After piecing together a few examples from books and online, Apply-Templates is making everything much easier. I realize that thinking of templates as objects makes a lot of sense (to parse this node, use this object – I mean template). Thinking this way produces a better design, which is explained more near the end of this article.

Two Types of Templates

Contrary to the introductory examples mentioned above, Apply-Templates is only part of the solution when building an application, and many XML/XSLT specifications are required in designing a larger application. The two basic concepts of XSLT are *calling templates by name* and *applying templates*.

- ***Calling templates:*** A fairly straightforward technique in which a template is named and then called to parse the selected data. When calling a template, parameters can be passed and these values can be used later to filter by or as another piece of information for the output.
- ***Applying templates:*** Relies on "matching" elements found in the data, then parsing each node – actually performing two operations.

Applications that use both types are much easier to design than trying to make everything conform to one technique or the other.

Analyzing an XML Document

Understanding how an application needs to be developed and the structure of the XML document are the most important aspects of designing an XML-based program. Begin by looking for parts of the data that are unique. After analyzing a document, you can break the content or data into sections. Less frequent data, such as author information for a book or a ship-

ping type for an order, can be parsed by calling templates by name, and the more repetitive data, such as line items or paragraphs, can be parsed using Apply-Templates.

Templates called by name are similar to functions in a script file or object class (e.g., getShipping, showOrderDetail). Templates that parse through many elements via template matching can be thought of as internal objects or collections. Match templates don't have names, but the element(s) they're matching take the place of a name. An example like <xsl:template match="lineitems"> can be thought of as the "line item" template.

"Understanding how an application needs to be developed and the structure of the XML document are the most important aspects of designing an XML-based program. Begin by looking for parts of the data that are unique"

Using Apply-Templates

When possible, working with Apply-Templates is a much cleaner approach than using XSLT programming logic. Logic will be implemented inside each element on a smaller scale; program flow logic can be limited to one or two conditional statements (or done away with altogether). More testing is needed to get each template matching and behaving properly, especially with mixed-content elements. In the end, however, you walk away with cleaner, more manageable, and more portable/reusable code.

As mentioned above, Apply-Templates consists of two actions: matching elements in the XML document and parsing elements to the output. A root template, <xsl:template match="/">, is created first and selects the entire document for parsing. Within this root template a named template such as <xsl:template name="showSection"> is called, which in turn invokes xsl:Apply-Templates where the content should appear. Finally, templates that match elements, such as match="para", match="title", or match="ulink", are created.

Matching can be simple, as indicated above, or more complex. You can match elements just with text, from a root element, and more. <xsl:template match="literal"> matches the "literal" element; <xsl:template match ="text()"> matches all text-type nodes within an element; <xsl:template match= "*"/> matches all elements; <xsl:template match="*|text ()"> matches all elements or any text nodes. Combining the use of calling templates, passing parameters, and using XPath queries makes matching less complicated.

Applying XPath queries to the Apply-Templates call itself is a powerful technique in parsing the correct content. To parse today's orders, the programmer can quickly create an XPath query and Apply-Templates to the subset, such as <xsl:Apply-Templates select="orders/[@orderdate=$date]". This is much easier than working within the template itself (<xsl:template match="orders/[some kind of matching selection]">). This code isn't reusable or hard-coded, and it's difficult to see if your match is valid. Using an XPath query from the select="" attribute when invoking Apply-Templates abstracts the query from the matching templates, making testing much easier. This is explained in greater detail below and in the listings.

A simple, more complete example is a matching template for a person's address. Since this template matches an <address> element, other elements that include addresses (such as businesses or orders) can be parsed using the same template. The application can have two "named" templates that call for person and business elements to be parsed. Both would invoke Apply-Templates to parse any address elements – for example, <xsl:apply-templates select="./person">. The main template matches an address and uses Apply-Templates to invoke the templates for "street1","street2","zip", etc. These templates can be reused later by invoking different select queries such as xsl:apply-templates select= "./orders">. Another way to accomplish the same task would be to include

"Apply-Templates consists of two actions: matching elements in the XML document and parsing elements to the output. A root template, <xsl:template match="/">, is created first and selects the entire document for parsing"

"Apply-Templates allows implementation of the elements needed now and the addition of templates for other elements later without worrying about existing code or structures"

on a Web site. However, programming a large number of elements from the DocBook DTD could make logical coding for each element a nightmare.

Apply-Templates allows implementation of the elements needed now and the addition of templates for other elements later without worrying about existing code or structures. There were still many questions on this project about how and when to select content from the document. After more analysis – deciding how the Web site should work and how the DocBook DOM object should be constructed – we built some "stub" templates – templates without any functionality – just as if we were starting to build an object-oriented class. Test strings (output) were placed into the templates to show whether the template was being called or matched correctly.

Using the one-template-fits-all approach, one template matching all elements in the document could have been used, or all paragraphs (para elements) could have been selected and checked for different kinds of content with logic such as "If there is an imageobject here do this". However, if new elements such as ulinks or titles needed to be parsed and weren't in the original code, more "if" or "when" statements would continue to be added. This wouldn't work well because more and more logic would be added to the template using (possibly nested) <xsl:if> or <xsl:choose> statements to navigate through the various elements of a DocBook. The DocBook specs are rather lengthy, and the single XSL template would start looking like some old, old Pascal code from the past.

the "address" element in the query. Starting with <xsl:apply-templates select="person/address">, the subelements will be parsed all at once. In either case, the next logical step is to place the address, street1,street2,zip, etc., templates in an address.xsl file and include it with any application that needs to parse standard addresses. Within this file the match="address" template can be used or not, giving the developer the option of using the templates either way (./person or ./person/address).

The way my application ended up working was:

1. Find the condition with an xsl:if or xsl:choose.
2. Call a template by name.
3. Load the proper XML data (optional if already loaded).
4. Use Apply-Templates to select and parse the data.

Analyzing the Document

I used the DocBook Article XML file found in Listing 1 and the DocBook Simple Book XML file in Listing 2. DocBook XML is a popular standard for technical and other documentation and the program needed to display content from DocBook Articles and a small DocBook Book

"The easiest way to select content is to provide an XPath query in the xsl:apply-templates function itself with a select attribute. The parameter "$page" is set from a JSP page when the file is loaded"

How We Used Apply-Templates

First I created a generic template (see Listing 3) that more or less displays the Web page in its simplest form (a table with two cells), calling the templates in Listing 4 by name. The showSection template takes a parameter of the current book (all the XML from the root node) and uses Apply-Templates to parse out the selected section. In the test application the book's content is loaded dynamically and put into this variable. However, if a template is parsed directly against a DocBook XML file, you could just use "/book/chapter/sect1[@id=$page]" in the select statement instead of "$workCopy/book/chapter/sect1[@id=$page]".

The select statements above prove again that the easiest way to select content is to provide an XPath query in the xsl:apply-templates function itself with a select attribute. The parameter "$page" is set from a JSP page when the file is loaded. A parameter for the chapter being viewed could also be added and the XPath would be <xsl:apply-templates select="/book/chapter[@id=$chapter]/sect1[@id=$page]">. Changing the select statement doesn't affect the templates.

One of the challenges facing the project was that only one section of the DocBook document needed to be parsed while displaying a list of pages for navigation. Most of the examples explain XPath queries separately from Apply-Templates. Selecting a good starting point with the correct XPath query made things much easier.

Another challenge was parsing mixed-element types (elements with text and subnodes) such as the <para> element, which can contain almost anything, including several text() nodes. This could have been avoided if the text was another subnode under the paragraph element (para/content instead of para/text()), but again, the DocBook DTD doesn't allow addition of elements such as this. In real documents there may be an inline image or link inside the text content as well (see Listing 6 for an example).

In Listing 4 the showArticle template filters a DocBook ARTICLE XML file and selects the entire article for parsing. The showSection template needs to parse all the paragraphs and subelements in the selected section of a DocBook BOOK XML file. The DocBook BOOK DTD is quite a bit different from the ARTICLE DTD, but Book Sections and Articles are similar. The code in Listing 5 shows how the application uses the same templates for elements containing text, links, and pictures in DocBooks or Articles.

Notice again that both these templates pass the XML tree as a parameter, and templates are applied only to the selected section of the DocBook file or to the entire article (Listing 4). In the final application the DocBook is loaded on every page, not just when a section needs to be viewed

CTIA Wireless 2002

www.ctiashow.com

because the navigation relies on the DocBook content. In the future Xlink or Xinclude to facilitate the dynamic inclusion of content may work better, but as long as the DocBook is small, everything is fine as it is.

Listing 5 is the workhorse that outputs the content. For now, these "matching" templates are located in the same file as the showArticle and showSection templates, but will be moved later into another included XSL file (outputelements.xsl). The last template in the file (match="para") processes the Article and Book section paragraph tags. Inside each paragraph element text(), titles, cite titles, links, and pictures are parsed and displayed.

Mixed types can cause paragraphs to contain multiple text() nodes, which can cause some problems since other elements containing text() nodes shouldn't be displayed. Using template matching, inline images, other media, links, and nodes with mixed content can be maintained as individual entities.

Listing 6 is an example of a mixed-content node. A matching text() template is needed because we're already matching the para node in the document and need to select Text(), Links, and other entities (objects). With the mixed-content node in Listing 6, the text() template gets called twice – once for the text before the <ulink> and <imageobject> tags and once after. Notice also that the normalize-space() function is used to properly display preformatted content or content in HTML text areas.

In our application the header information in articles, <artheader>, shouldn't be parsed and viewed. By default, the template match="text()" will display all the text found in the artheader elements as well. The "artheader" template overrides the "text()" template and the content is not displayed. With the current set of templates the Article Header information won't show up, but in the future a template named "showArticleInfo" could be called before or after the showSection or showArticle templates are called.

When invoking Apply-Templates, we could have included paragraph elements in the select attribute (/book/chapter /sect1[@id=$id]/para – see Listing 4). However, DocBook also allows for other elements on the same level as paragraphs. The Program Listing element, <programlisting>, and Formal Paragraph element, <formalpara>, are widely used and definitely could be implemented later, so it's better that the main template parse sections and articles. In either case the templates matching pictures, text, and links will be separated into their own XSL file to be reused later.

Thoughts About Designing the Application with Apply-Templates

To wrap things up, I'm starting to think about templates as object classes rather than as strict XSL code and code functions. If done correctly, the templates, just like objects, lend themselves to be reused and will organize your data and logic in a much more structured and coherent fashion than if you were to write hundreds of lines of code. Near the end of developing this application I found myself saying, "I need an object to do this." Following this logic, an XSL file of related templates can be regarded as a package (such as the address.xsl file mentioned above). XSL files can be designed to take on different roles, such as templates that call other templates, templates that have business logic or calculations, and, finally, templates that parse elements.

On a larger project someone else could develop templates that parse a new set of elements (such as a calendar or product elements) in another file that is included later. Some templates will have logic code statements such as <xsl:if> and <xsl:choose>. However, limiting the roles of different types of templates and using Apply-Templates with XPath queries instead leaves room for expansion without worrying about changing other parts of the code.

Author Bio

Roy Hoobler has been involved with Internet programming since 1995 and developing Web businesses and "niche" market sites since 1996. In 1998 he joined Net@Work Inc. (www.netatwork.com) where he designs (and still codes) business and cutting-edge e-commerce applications using a wide range of technologies, including XML. Roy is MCSD certified.

RHOOBLER@NETATWORK.COM

LISTING 1 DocBook article example

```
<article><title>Preparing to be Scalable...</title>
<artheader>
        <author>
                <firstname>Roy</firstname>
                <surname>Hoobler</surname>
        </author>
</artheader>
<para><citetitle>Planning for a lot of visitors</citetitle>
 Running a Web site is a lot of work and it is nice to know
 that things will...
</para>
<para><citetitle>Use good database tools.</citetitle>
 Database connections are by far the most intensive
 operations on your site...
</para>
</article>
```

LISTING 2 Simple DocBook example

```
<book>
<title>mysite</title>
 <chapter><title>Chapter1</title>
 <chapterinfo></chapterinfo>
   <sect1>
  <para><citetitle>a title</citetitle>
   first paragraph
  </para>
  <para>
   <ulink url="link.jsp">my link</ulink> a link
   more text for 2nd paragraph
  </para>
  <para>
   <inlinemediaobject>
    <imageobject>
     <imagedata fileref="myfile.gif"/>
    </imageobject>
   </inlinemediaobject>
   <ulink url="mypicfiles.jsp">more pictures</ulink>
  </para>
   </sect1>
 </chapter>
</book>
```

LISTING 3 Top-level XSL sheet: show either article or book section

```
<xsl:template name="showSomething">
 <table><tr><td>My Navigation Bar</td>
 <td>
 <xsl:if test="$topic != '' ">
  <xsl:variable name="article" select=
   "document(concat($filesDirectory,'/p/',$post,'.xml'))"/>
  <xsl:call-template name="showArticle">
   <xsl:with-param name="workCopy" select="$article"/>
  </xsl:call-template>
 </xsl:if>
 <xsl:if test="$topic = '' ">
  <xsl:variable name="work"  select=
   "document(concat('../../',$filesDirectory,'/',$xmlfile))"/>
  <xsl:call-template name="showSection">
   <xsl:with-param name="workCopy" select="$work"/>
 </xsl:call-template>
 </xsl:if>
 </td></tr></table>
</xsl:template
```

LISTING 4 Using named templates to invoke Apply-Templates on different elements

```
<xsl:template name="showArticle">
 <xsl:param name="workCopy"/>
 <xsl:apply-templates select="$workCopy/article"/>
</xsl:template>

<xsl:template name="showSection">
 <xsl:param name="workCopy"/>
 <xsl:apply-templates select=
   "$workCopy/book/chapter/sect1[@id=$page]"/>
</xsl:template>
```

LISTING 5 Matching templates applied to our document

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

 <xsl:template match="artheader">
 </xsl:template>
 <xsl:template match="programlisting">
  <pre>
  <xsl:value-of select="normalize-space(text())"/>
  </pre>
 </xsl:template>
 <xsl:template match="title">
  <b><xsl:value-of select="text()"/></b><br/>
 </xsl:template>

 <xsl:template match="citetitle">
  <b><xsl:value-of select="text()"/></b><br/>
 </xsl:template>

 <xsl:template match="ulink">
  <a href="{@url}"><xsl:value-of select="text()"/></a> 
 </xsl:template>

 <xsl:template match="inlinemediaobject">
  <img src="{./imageobject/imagedata/@fileref}"
   border="0" align="right"/>
 </xsl:template>

 <xsl:template match="text()">
  <xsl:value-of select="normalize-space(.)"/> 
 </xsl:template>

 <xsl:template match="para">
  <p>
  <xsl:apply-templates/>
  </p>
 </xsl:template>
</xsl:stylesheet>
```

LISTING 6 Mixed-content node with an image

```
<para><citetitle>Portal Heaven</citetitle>
        normal text
 The picture and a <ulink url='test.jpg'>link</ulink>
 <inlinemediaobject>
 <imageobject>
 <imagedata fileref="http://rch.site.com/images/mypic.jpg"/>
 </imageobject>
 </inlinemediaobject>Another <ulink url=
   "http://www.site.com/dir/">Portal</ulink>
</para>
```

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

q&a

[WRITTEN BY TRACE GALLOWAY

Questions and answers from Infoteria's 'Ask the Xperts'

DiveintoXML

For this month's column I thought I'd share some of the real-world questions we've received over the past year from our "Ask the Xperts" forum on Infoteria's Web site (www.infoteria.com). These problems range from XSLT recursion, B2B implementations, DOM, SAX, JDOM, and UDDI to Web services using XML technologies. I've included some of my favorites from the thousands of questions we've received.

Q: ***I'm transforming XML data to HTML. When I use Infoteria's iXSLT to do transformations, it leaves the XML declaration string. Why?***

A: By not specifying the output method using <xsl:output method="html"/>, you're telling the processor to output XML by default. In accordance with the XSLT 1.0 Recommendation, an XSLT processor is required to provide the XML declaration in the output if it's provided in the input XML source. The following is from the XSLT 1.0 Recommendation:

The default for the method attribute is chosen as follows. If

- the root node of the result tree has an element child,
- the expanded name of the first element child of the root node (i.e., the document element) of the result tree has local part html (in any combination of upper and lower case) and a null namespace URI,
- and any text nodes preceding the first element child of the root node of the result tree contain only whitespace characters,

then the default output method is html; otherwise, the default output method is xml. The default output method should be used if there are no xsl:output elements or if none of the xsl:output elements specifies a value for the method attribute.

The foregoing rule also includes the absence of the <xsl:output> altogether. In the stylesheet you've written (not included in this article), if you look at the output document, you'll see that your result tree doesn't meet the requirements outlined above. To be honest, although I've read the Recommendation several times, I've always felt the authors were somewhat ambiguous in this area.

The "and" at the beginning of the third requirement leads one to believe that either the first rule must be met *or* the second and third rules must be met. In reality, all three must be met for the default output method to be HTML. When transforming from XML to XML, use the <omit-xml-declaration="yes"/> attribute in your stylesheet to achieve the same result.

Q: ***I'd like to convert four AS/400 DB2 databases into XML documents so I can send (DB2 AS/400 – Oracle mapping) the XML docs to an Oracle DB in another department. I'd also like to do the same in the other direction: Oracle —> AS/400 DB2. Is this possible?***

A: Producing XML documents from a DB2 database and then writing those documents into an Oracle database is precisely what Infoteria's iCONNECTOR products are designed to do. iCONNECTOR is a bidirectional tool that allows you to create XML from databases and write XML data into databases. The iRULEGENERATOR graphical mapping utility is included with all of the iCONNECTOR products and it provides a user-friendly environment that allows users to graphically define relationships between their XML data structures and database tables and fields. In addition, these products are designed to run on Windows NT/2000 and can be run as a Windows service on either platform. All of the iCONNECTOR products provide both a Java and COM API through which C++, Java, VB, or other client applications can access the iCONNECTOR service functionality. The products don't need to be run on the same machine as the database, and one iCONNECTOR (per database type) can access any number of databases. You mentioned that you have four DB2 databases; one iCONNECTOR for DB2 can be used to access the information stored in all of them.

Q: ***We want to use XML for EDI. We have some experience in EDIfact, but a customer wants to send us XML-based data, which we need to transfer into our IBM AS/400 DB2 database. How might we receive/send XML documents and bring them into our database?***

A: You might use a component like Infoteria's iMESSENGER, which is designed to monitor a POP3 or iMAP4 mailbox looking for XML data as either a message attachment or as well-formed XML directly in the body field of the message. Once received, iMESSENGER extracts the XML and writes it into a working directory. From here iMESSENGER can be configured to initiate an internal process to write the information into your database. That internal process could be iCONNECTOR, a fully functional 30-day trial version, which can be downloaded from the Web site.

AUTHOR BIO

When he's not diving for lobsters, Trace Galloway is the director of field engineering at Infoteria Corporation (www.infoteria.com), an XML software development company based in Tokyo and Beverly, MA. An expert in XML and XSLT technologies, Trace specializes in RosettaNet B2B implementations.

Q: ***Is there a way to generate XML from a DTD using some type of an editor?***

A: Absolutely. XML Spy from Altova Corporation allows you to create a new XML document based on an existing DTD or schema. XML Spy is without a doubt the preferred XML/XSL editor when working in the XML environment. You can download a free trial version at www.xmlspy.com/download.html. After downloading the install program, run through the setup routine and you'll come to the components download portion of setup. When you choose to install additional components, install Infoteria's iXSLT as your default processor for XSL Transformations. Once you've set up the program, run it and choose new XML document from the file menu. You'll be prompted with a dialog box that asks if you want to use an existing schema or DTD as the basis for your new XML document.

Q: ***I'm evaluating the iXSLT product and so far I'm very impressed. I have one question regarding your support for ID/IDREF: Is there any?***

A: Yes, with the following notes: (1) If the ID attribute has been defined in an external DTD subset, remember to use the -g or -G (iXSLT command line switches) to ensure that the processor reads in the external DTD; (2) iXSLT 2.0c supports xsl:key and it may be used when necessary – see the XSLT 1.0 REC for complete details; (3) when using ID, IDREF, and IDREFS, refer to section 12.2 of the XSLT Recommendation (and accompanying specifications) for proper usage. The XSLT 1.0 REC can be found at www.w3.org/TR/xslt (a full discussion of xsl:key, ID, IDREF, and IDREFS is outside the scope of this Q&A article. I hope to cover that topic for ***XML-Journal*** in the future).

Q: ***Why doesn't your iXSLT processor convert the XML closing tags from <br/> to the standard HTML
?***

A: When an XSLT processor creates the result tree as an XML document, conforming to the XSLT Recommendation's section 16.2 HTML Output Method isn't required.

The HTML output method should not output an end tag for empty elements. For HTML 4.0, the empty elements are area, base, basefont, br, col, frame, hr, img, input, isindex, link, meta and param. For example, an element written as <br/> or
</br> in the stylesheet should be output as <br>.

Use the <xsl:output method="html"/> and the processor will recognize the names of HTML elements regardless of case. For example, elements named br, BR, or Br should all be recognized as the HTML br element and output without an end tag. If you include the <xsl:output method="html"/> element in your stylesheet, you'll see that the <br/> elements in your stylesheet do get converted into <br> elements in your HTML result. In addition, the XML declaration will no longer be present.

Q: ***My current back-end system runs on an MS Access 2000 DB. One of my largest suppliers has requested that we implement an electronic invoicing system based on the RosettaNet framework. What do I need to do to convert my current paper system to an electronic RosettaNet–based one?***

A: Infoteria offers iCONNECTOR for MS Access and Asteria Server for RosettaNet. The former allows you to extract the relevant data from your Access DB and move it into the appropriate fields in a RosettaNet-compliant XML document (the "Service Content" portion of your RosettaNet message). From there it's a simple, straightforward process to map your data between the two structures.

Once you've produced your PIP 3C3 XML document (essentially, your "electronic" invoice), you have a couple of choices. First, you could develop a process consisting of one or more applications that converts the XML document you produce from the DB into a truly compliant RosettaNet Object (RNO) or RosettaNet Business Message (RBM). This "process" would at a minimum need to include functionality to create the required additional headers (Preamble, Service Header, and possibly a Delivery Header), a mechanism to validate your XML document against the PIP 3C3 DTD, and the PIP 3C3 RosettaNet Message Guideline. It would also need to add binary length information to the resulting document, include attachments inside your resulting RNO or RBM, and potentially provide a transport mechanism to send the document to your supplier.

The B2B server Asteria includes the functionality listed above as well as scheduling services, multiple transport protocol support, GUI-based tools to allow administrators to configure TPAs between partners, scripting facilities, and many others. It's available in a variety of configurations.

TRACE@INFOTERIA.COM

xmatters

[WRITTEN BY COCO JAENICKE

Some metrics for how a native XML database ought to act

Going Native

As XML is put into production more frequently, many of the XML software product categories are beginning to gain clarity. One area that has seen several new entries is the XML database management system, or XDBMS.

The word *native* is frequently used when describing XML database management systems. (It seems that the only adjective used more freely is *first*. And just for the record, the first three ship dates I found are eXcelon 3/1999, Software AG 10/1999, and IXIASOFT 1/2000 – all claiming to be first.) *Native* can have several different meanings, and it doesn't necessarily bestow instant special status to any of these products.

The term *native XML database* denotes that XML data is stored as XML. That's a bit vague and meaningless, of course, as a BLOB or a flat file technically qualifies. Rather than argue the semantics of the "right" or "proper" way that XML is meant to be stored, it's more productive to examine some metrics for how it ought to act.

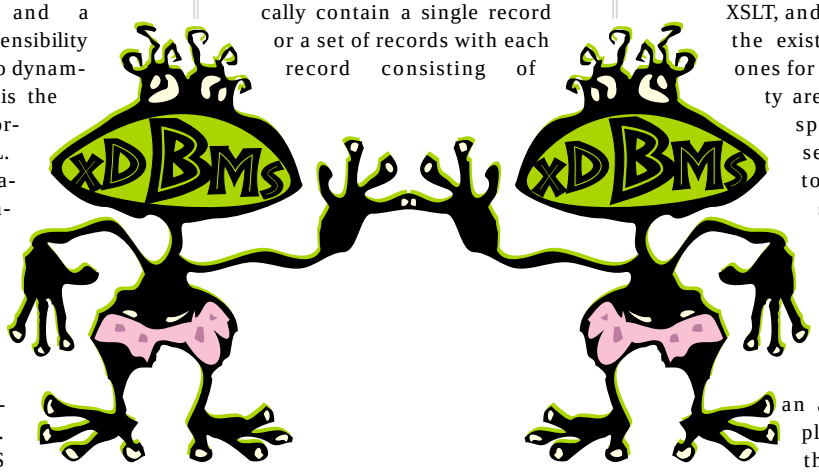
At a minimum, a native XML database should:

- ***Support XML extensibility:*** This is probably the most important distinguishing characteristic between a native XDBMS and a wannabe. The extensibility and the ability to do dynamic data modeling is the single most important facet of XML. Without it, infrastructures are as immobile as ever, with even the simplest change in partners or system requiring a schema evolution and application rewrites. A native XDBMS should be able to accept an XML document with an extended schema (additional elements) without touching existing records or requiring any existing applications to go offline.

- ***Store arbitrary XML:*** Along the same lines, a native XDBMS should be able to accept arbitrary XML documents (well-formed XML with no DTD) without difficulty. Should the document contain an unexpected structure (either additional, missing, or rearranged elements), the database should be able to store and manage it natively, without awkward workarounds. This enables IT managers to design and build infrastructure without having prior knowledge of every data transaction.

- ***Create, read, update, and delete documents, fragments, and elements:*** While XML documents can be of any size or level of granularity, they typically contain a single record or a set of records with each record consisting of many elements. The level of granularity used by the database may be different from the level desired by the application, and in fact multiple applications may access the XML data, each preferring to slice and dice the information in different ways. For this reason the database should be able to serve individual elements, fragments, or the entire document. Otherwise, the application developer is left with the task of parsing and reconstructing XML. Even if there are client-side utilities to do this, it requires large amounts of memory, which quickly becomes unwieldy as the document size grows. Even worse, the client has to fetch the document from the server and reparse for every request – or be content with dirty reads.

- ***Support all XML standards:*** XML is about more than standardizing a metalanguage – the W3C has also ratified (and continues to ratify) many standards for accessing and manipulating data as well. XPath, XSLT, and XQuery are some of the existing standards, and ones for updates and security are on the way. These specifications represent the native way to access data and should be supported by the XDBMS. Retrofitting a SQL command or asking the developer to learn a proprietary interface is an alien approach and places the burden on the developer.



AUTHOR BIO

Coco Jaenicke was, until recently, the XML evangelist and director of product marketing for eXcelon Corporation. She played a key role in the successful development and introduction of eXcelon, the industry's first application development environment for building and deploying e-business applications. Coco is currently an independent consultant.

> *Native* can have several different meanings, and it doesn't necessarily bestow instant special status to any products

- ***Preserve document content and structure:*** Data integrity is almost too obvious to mention but, given several nonnative approaches to managing XML data, can't be omitted. This requirement – often referred to as *round-tripping* – states that when you put data in and then go to retrieve it, the result is identical to what you started with. While this is something we take for granted when working directly with a relational database, we can't assume it to be true (yet) for XML databases. If the underlying storage model isn't native XML, a certain amount of translation that can introduce errors occurs when storing and retrieving – especially if the structure is complex.

These metrics can be applied to several currently accepted implementation models for XML management – some of which are more native than others: flat files, a relational database with adapters, persistent DOM, or some hybrid thereof.

Storing an XML document in a flat file ensures all the flexibility needed to support an extensible or arbitrary schema, but doesn't support any of the access or manipulation requirements. For most production environments flat files aren't an acceptable data management solution for many reasons. Dropping the document into a relational database as a BLOB adds a few "-ilities" to the picture, but building applications with XML is still difficult and time consuming. All parsing, searching, updates, transformations, and so on are left as an exercise for the programmer.

Managing XML in a relational database has the upside of being legacy-friendly and requires just a simple add-on to existing systems. However, such an approach relies on a wrapper to provide the XML features described previously. This works well when only some of the requirements of an XDBMS are needed, but meeting all the metrics listed above is difficult for a wrapper and performance can degrade rapidly.

Storing and managing XML as a persistent DOM involves parsing the data on the way in and storing the elements individually as hierarchically linked nodes. This model supports extensibility and arbitrary schemas because individual elements can be added or deleted at will, and elements or fragments can be updated or accessed without programmers having to reparse the whole structure. While this model is the most native at the core, it is also the furthest from comfortable, traditional models.

There are also many hybrid models that combine several of the above approaches to try and hit as many features as possible. Hybrids can solve many problems, but they certainly do stretch the definition of *native*. They are a clever collection of band-aids, and we can expect that there will always be devils in the details.

XML databases come in many different varieties. The functionality varies widely, and we are easily lulled into complacency because of our familiarity with relational databases. The word *native* deserves extra examination because it is widely – and often erroneously – used, and is too often treated merely as a checklist item. An XML database may one day be a commodity, but for now a careful assessment of features and needs is the prudent course.

CJAENICKE@MEDIAONE.NET

> The word *native* deserves extra examination because it is widely – and often erroneously – used, and is too often treated merely as a checklist item.

[WRITTEN BY SHOUKI SOURI

How to transform XML to SVG easily

GotXSLT?

Part 4 of 5

The two most commonly used graphics formats on the Web today are GIF and JPEG. Both are pixel-based and, as we've all probably seen, lose quality when you try to zoom. SVG – Scalable Vector Graphics – is entirely based on XML, and takes up less space than the average GIF and JPEG.

SVG brings to the Web the rich, high-resolution graphics that we've all come to expect in printed catalogs and magazines. And because of its vector-based nature, it has the same high quality when it's displayed on PDAs with limited screen real-estate as when it's displayed on high-end monitors (even TVs), or when it's printed out on paper.

In previous issues of ***XML-J*** (Vol. 2, issues 10, 11; Vol 3, issue 1) I've offered instruction in how to transform XML documents into HTML, WML, and VoiceXML. In this tutorial I'll give you a basic introduction to SVG and show you how to use XSLT to transform our XML example into SVG.

Note: Currently, to view SVG graphics, you need to download and install a compatible viewer or Web browser plug-in. Check the **Resources** section for a list of available programs. I've used Adobe SVG Viewer (3.0) for Windows – English, and a Xalan-java version XSLT processor.

What Is SVG?

SVG is a powerful language for describing two-dimensional vector and mixed vector/raster graphics in XML. It is defined best perhaps by W3C as

> a language for describing two-dimensional graphics in XML. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images, and text. Graphical objects can be grouped, styled, transformed, and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, and template objects.

The SVG 1.0 specification was released in September 2000 as a Web standard (a W3C Recommendation). The specification defines the features and syntax for SVG, the result of substantial implementation experience with generators, viewers, and transcoders based on SVG and the amount of SVG content developed to date.

On October 30, 2001, the W3C SVG Working Group (WG) announced the first public release of the SVG 1.1 specification and also of two SVG 1.1 profiles, SVG Tiny and SVG Basic (collectively, Mobile SVG). This is an initial draft, which we won't be using for this tutorial.

Why SVG?

Most graphics on the Web today are in bitmap formats (e.g., JPEG, GIF, and PNG), and thus contain information about each pixel in the image. *Vector graphics* on the other hand describes only shapes and paths – a much more efficient approach. Unfortunately, until the arrival of SVG, we've only had proprietary vector graphic standards, such as Flash and QuickTime, and no Web standard.

SVG changes this. In addition to faster download speeds, SVG also comes with many end-user benefits. Some of the nice features:

- High-resolution printing
- High-performance zooming and panning inside of graphics without reloading
- Animation
- Filter
- Scripting
- Linking

This results in the user's being able to dig deep into an SVG object to obtain more dynamic information – a perfect application for various Internet business applications. An additional benefit is that since it's based on XML, it's entirely text-based, which will facilitate the ability of search engines to index SVG images; users will thus be able to search for text within images (e.g., search for a button text or a street name on a map). With its text-based nature, it's also possible to create SVG images on the fly (e.g., with a database back end).

In summary, SVG 1.0 had many key benefits, including:

- Small file sizes for faster Web page downloads
- Unlimited color and font choices
- Zoomable graphics and images – ideal for maps
- Scripting control for custom interactive events and animation
- Clean, crisp, high-resolution printing from Web browsers
- Bitmap-style filter effects for high-impact graphics
- Easy integration of text-based format with other Web technologies
- Support for Cascading StyleSheet (CSS) and other devices such as palmtop computers.

Recalling Our XML Document

I hope that you still remember our XML example (mybooks.xml). To refresh your memory, Listing 1 depicts our XML file. (Listings 1-4 can be found online at www.sys-con.com/xml/sourcec.cfm). As explained in my previous article, you're free to modify your XML document any way you like, but then you have to pay extra attention to some of the details when we get into creating our SVG stylesheet.

The Goal

Our goal is to produce an SVG Web page that looks like Figure 1.

Understanding Our SVG StyleSheet

We'll go in with some details over the transformed SVG file. SVG code is relatively straightforward. Once we get ourselves exposed to the syntax, the rest is as easy as reading text.

AUTHOR BIO

Shouki Souri, 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.

Note: SVG files are composed of plain text. You can examine the source coding by placing the mouse cursor over the image, clicking the right mouse button, and choosing the option View Source.

Step 1: SVG file skeleton

Because it's an XML file, a SVG file has to have a precise structure. The following code serves as a template for any SVG file you would create from scratch.

```
<?xml version="1.0" encoding=
 "ISO-8859-1"?>
<svg xml:space="preserve" width=
 "450" height="300" >
<desc><!-- put a description here --
 ></desc>

</svg>
```

The following SVG code fragment defines the width and height of the SVG drawing area in pixels:

```
 <svg xml:space="preserve" width=
  "450" height="300">
```

Since the XML language requires that all tags must be closed, the SVG file must end with:

```
</svg>
```

Step 2: Add a title

The following code fragment shows the simplest text command:

```
<text x="100" y="100">My Book Catalog
 </text>
```

This text will have the default color and stroke characteristics without our formatting style.

The following code illustrates how to format (style) the text and also how to position the annotation fragments:

```
<text style="font-size:20;
 text-anchor:middle" x="210" y="260"
 fill="blue" id="title">
 My Book Catalog</text>
```

Step 3: Create a drawing space (rectangle)

The following code segment shows how to create rectangles. First we create a gray drop shadow rectangle. We want it three units to the right and below the front rectangle. We do this by using the transforming function to translate the rectangle by three units in both dimensions.

```
<rect stroke-width="4" stroke=
 "#999999" fill="#999999" transform=
 "translate(3, 3)" x="38" y="20"
 width="360" height="200"/>
<rect x="38" y="20" width="360"
 height="200" style="fill:#E3DFCC;"/>
```

Step 4: Create a frame and grid lines

The following code segments first create a frame and then draw horizontal grid lines using path command. Paths are described using the following data attributes: "moveto'" (set a new current point), "lineto" (draw a straight line), and "closepath" (close the current path by drawing a line to the last "moveto" point). The following code specifies a path in the shape of a L frame. The "M" is a shortcut for "moveto," "L"s are for "lineto"s, and the "z" is for a "closepath."

```
<g style="stroke-width:2; stroke:black">
        <path d="M 40 220 L  40 20 L
         40 220 L 200 220 Z"/>
 </g>
```

The following code segment is responsible for drawing the horizontal grid lines. Note that we have grouped all the path elements using the <g> and </g> tags. This in turn can make all the graphical elements within these tags share a common identity and style.

```
<g style="fill:none; stroke:
 #B0B0B0; stroke-width:1; stroke-
 dasharray:2 4">
    <path d="M 42 200 L 198 200 Z"/>
    <path d="M 42 180 L 198 180 Z"/>
    <path d="M 42 160 L 198 160 Z"/>
    <path d="M 42 140 L 198 140 Z"/>
    <path d="M 42 120 L 198 120 Z"/>
    <path d="M 42 100 L 198 100 Z"/>
    <path d="M 42  80 L 198 80 Z"/>
    <path d="M 42  60 L 198 60 Z"/>
    <path d="M 42  40 L 198 40 Z"/>
    <path d="M 42  20 L 198 20 Z"/>
</g>
```

Step 5: Create bars – (red)

Similarly, we will use the Path command to draw the bar column. And since all the bars are drawn the same, we will only discuss the red bar. As indicated, this bar is filled with a solid red color.

```
<path style="stroke-width:2;
 stroke:black; fill:red" d="M 50 220
 L 50 80.2 L 70 80.2 L 70 220 Z"/>
```

By making opacity less than 1.0 (see below) we can create a lighter see-through red color instead. If the opacity command is omitted or set to 1.0, the fill will be solid and opaque as in our case above.

```
<path style="stroke-width:2;
 stroke:black; fill:red; opacity:
 0.60" d="M 50 220 L 50 80.2 L 70
 80.2 L 70 220 Z"/>
```

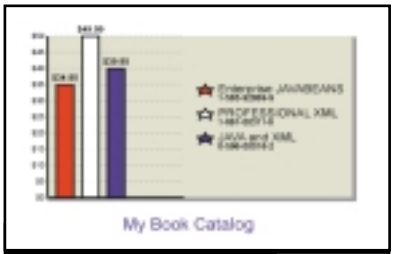


FIGURE 1 Our goal SVG Web page

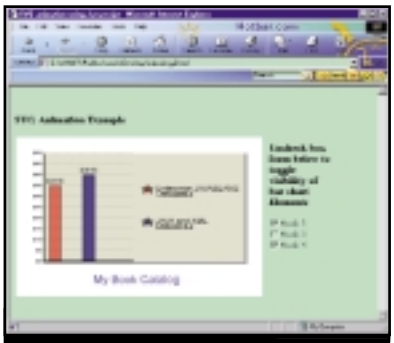


FIGURE 2 SVG animation using JavaScript

The same thing applies for the rest of the bars.

Step 6: Create text legends and stars

This final step is responsible for creating a legend text for the book's name and ISBN number. The stars are created using the path command.

```
<text style="font-size:10; text-
 anchor:middle" x="40"
 y="75.2">$34.95</text>
<text x="240" style="font-size:14;
 text-anchor:start" y="90">Enterprise
 JAVABEANS</text>
<text x="240" style="font-size:10;
 text-anchor:start" y="100">1-565-
 92869-9</text>
<path style="stroke:black; stroke- width:2;
 fill:red" d="M 225 80 L 222  85 L 215
 85 L 220 88 L 218 93 L 225  90 L 232 93
 L 230 88 L 235 85 L 228  85 Z"/>
```

Performing Transformations from the Command Line

At the very minimum, you must include xalan.jar and xerces.jar on your system class path to perform the translation. I have saved both mybooks.xml and mybook2svg.xsl (see Listing 2) under gotxslt directory. To perform the translation, type the following:

```
D:\>cd gotxslt
D:\gotxslt>java
org.apache.xalan.xslt.Process -in
mybooks.xml -xsl mybooks2svg.xsl -out
mybooksinsvg.svg
```

If all goes well, the new output file mybooksinsvg.svg (see Listing 3) will be the resultant output from XML to SVG translation. This a pure ASCII file, so you can open it and study it in any text editor.

Advanced Features of SVG

It's the advanced features, including effects, animation, and scripting, that make SVG worthwhile. Effects such as blurs and filters enable designers to create vector graphic images permitting the production of Web artwork that expedites client-side generation and alteration. This means that content for your vector artwork can be generated dynamically and then rendered with sophisticated filter effects. Animation occurs on a per object basis. Fading, growing, and rotating objects is possible, and timing can also be precisely controlled. Yet sometimes it's not possible to produce the desired animation using the declarative animation elements defined in the SVG standard. For instance, you might want to animate a nonscalar attribute of an SVG element, or toggle visibility using check boxes or radio buttons. In these cases you can still create the effect using the combination of JavaScript and DOM access to the SVG document.

Figure 2 is an example of a JavaScript animation, which includes check boxes to toggle using a mouse click, on or off, the visibility of the bar elements (books). Listing 4 shows a complete JavaScript and HTML and how to embed them into an SVG file.

Conclusion

This article has shown a way of using XML and XSL Transformation to create an SVG file.

Although the SVG examples presented here are simple, they highlight the importance of this emerging technology. I believe that we'll see SVG popping out all over the Web; however, I don't think we'll be working with hand-coding SVG. Instead we'll probably just be using the same tool we're using today to create our images. Nevertheless, we've shown that even if you have your data in XML and you want to present it in a graphical image such as SVG, XSLT is a great place to start.

Resources

- ***Adobe SVG Viewer:*** www.adobe.com/support/downloads/product.jsp?product=46&platform=Windows
- ***W3C SVG site:*** www.w3.org/TR/SVG/
- SVG implementations: www.w3.org/Graphics/SVG/SVG-Implementations
- ***Adobe SVG Tutorial (excellent introduction):*** www.adobe.com/svg/tutorial/intro.html

SSOURI @ PANAMSAT.COM

Read Full Text At www.sys-con.com/xml/newsletter1.cfm

Continued from page 8

Now if we're only interested in a single XML language for standard business forms, then we can simplify this approach by eliminating that journey out to the abstract layer and just define the data model right in the XML schema. As far as I know, no one has yet identified a data modeling requirement for electronic commerce that can't be met with XML schemas.

The answer to the question of when ebXML will arrive depends on what you mean by "ebXML." It's a big initiative with a lot of pieces. To me, the most interesting parts are the data dictionary and the infrastructure pieces. The data dictionary work is a continuing project that will probably take years to finish, and I expect UBL itself to feed back into that. But the infrastructure pieces are already done. Version 1.0 of the ebXML specifications for secure XML messaging, trading partner agreements, and registries was completed in May 2001. These are standard electronic trading specifications of the United Nations, and they are usable right now. Work on all three specifications is continuing in OASIS, and revised versions are already nearing completion.

Thanks to ebXML, we've now got secure XML messaging built on SOAP; we've got a consensus on how to form trading partner agreements, which can be done either manually or automatically; we've got the basic specification for a very powerful taxonomy-driven registry; we've got a technology for the discovery and classification of core data components in the data dictionary; and we've got a preliminary understanding of how a library of components can be changed to reflect the current business context in which they're being used. If you add a standard syntax to the infrastructure pieces already defined by ebXML, we're ready to rock and roll. So even though the more visionary pieces of ebXML still have a long way to go, for projects over the next few years I personally consider ebXML version 1.0 essentially done, and what I want to see us do is start using it.

XML-J: ***Why is the OASIS process in particular so well suited, in your view, to a convergence/interoperability challenge like this?***

Bosak: Several reasons, actually. First, the OASIS process has a decentralized management model that makes it easy to get projects underway without a lot of overhead. Second, the process is completely open. All the mail lists are open to public view, and every technical committee has a mail list for public comment that anyone can subscribe to. And third, anyone can join OASIS. It's not limited to big companies. I think that this is essential if what we're designing is going to be accepted as an international business standard.

XML-J: ***When organizing and leading the original W3C working group that created XML, had you any idea that it would spread through the verticals with the speed that it has done?***

Bosak: I thought the adoption of XML would happen slowly but inevitably, the way grass grows up through the cracks in the pavement. XML is basically a disruptive, antimonopolistic technology like Java and Linux. I expected tremendous resistance for the first few years, especially from Microsoft.

And I think that's just what would have happened if it hadn't been for Jean Paoli, who was the Microsoft rep on the XML working group, and his boss at the time, Adam Bosworth. Adam focused on the data exchange possibilities of XML, which to the rest of us was just one small piece of it, and somehow he and Jean persuaded Microsoft to pick up that part of XML and run with it. That was about the last thing I expected to happen.

None of the uses to which XML is being put surprise me at all; everyone involved in the effort had a pretty clear idea of what you could do with XML. The big surprise was seeing Microsoft put its marketing machine behind it.

XML-J: ***Has your committee had any interaction yet with Microsoft in the process of developing this standard?***

Bosak: Microsoft sent an observer to one of our early organizational meetings, but no one since then. I think it would be really smart of them to be involved in UBL. But I guess you can't expect them to be that quick on the uptake every time.

XML-J: ***Will .NET have any impact – positive or negative – on UBL?***

Bosak: I don't know enough about .NET to answer that very well. It does seem to me that UBL would fit very nicely into the BizTalk framework, so I suspect that UBL would fit with .NET as well, but it's hard to be sure about that. In messaging terms, UBL is about the payload, not the wrapper. Or to put it in business process terms, it's about standardizing the documents, not the workflow. So my hunch is that UBL won't have any negative impact on .NET and might even help it. As to whether .NET will have an impact on UBL, that probably depends on whether Microsoft is developing a proprietary set of XML schemas for business documents. I don't think that would be a very sensible or productive thing for them to do, but you never know.

XML-J: ***Are there any plans for an East Coast meeting of the UBL TC any time in 2002?*** **Bosak:** The meeting schedule for the UBL TC is four times a year, twice a year on the West Coast of the U.S., once a year on the East Coast, and once a year in Europe. The first meeting of 2002 [was] hosted by Sun Microsystems 22–25 January in Menlo Park. The second meeting will be hosted by the UN/EDIFACT Working Group at their meeting 18–22 March in Barcelona. We haven't set definite dates and times for the third and fourth meetings yet....

But that's just the TC itself. We held our first TC meeting a few weeks after the attacks in September, and it was clear by that time that travel was going to be a big problem. So we structured our subcommittees so that they meet mostly by phone and e-mail. If people are interested in participating in the technical work, they shouldn't let travel stop them. They can find out about all this on the TC Web site at http://oasisopen.org/committees/ubl.

XML-J: ***How long will it be before the UBL transformation rules are formulated and then actually usable by the targeted businesses?***

Bosak: We've got one group of people working on the core library, another on the context methodology. We originally thought that the content definition would happen first and then we would attack the context methodology; if that's how it goes, I'd estimate about a year for each piece. But if the work can proceed in parallel at about the same pace in both groups, we should have a first generation of UBL available in about 18 months. We're hoping to have the very first document type, which will be a purchase order, out...sometime in February.

[WRITTEN BY NICK XIDIS

Improve custom branding while reducing development costs

Enhydra Design Patterns for ASPs

Currently Enhydra is receiving lots of attention for its capabilities as a server for wireless applications, but application service providers (ASPs) can use it to improve their ability to provide custom branding while at the same time reducing development costs. This article outlines some of the design patterns we used with Enhydra to create a set of applications that support thousands of very different looks from a single code base.

Even more impressive is the potential for evolution. Instead of using file-based development, as we did back in the days of C programming, we can begin to apply object-oriented concepts, like classification and abstraction, to HTML development. In C programming we divided up our code into files that corresponded to the coarse-grained concepts in the software we created. That's pretty much how Web developers divide up their HTML. With object orientation we can divide our program into classifications (classes), and classes are wonderful things because, unlike files, they can have complex relationships to other classes (association, inheritance, aggregations, etc…). Using XMLC and some well-known design patterns, you can apply OO principles to Web development, including inheritance.

The Requirement

Our client is an ASP for a group of about 30 commercial financial institutions. Each of which uses the products internally and resells them to thousands of retail financial institutions. The unique look each of these institutions has is a valuable commodity, and having a look and feel that consistently matches their branding is an absolute requirement.

Developing server applications that can support thousands of different looks based on a user's login is difficult. Before Enhydra, we were limited to the items available in Cascading Style Sheets (i.e., colors and fonts) and perhaps a different logo in the corner. Or we would end up with tons of JSP pages that required lots of maintenance and made making wholesale application changes very expensive.

The Bridge Pattern

The first pattern we used is the Bridge pattern (see Figure 1) from the "Gang of Four" book (*Design Patterns*, Gamma et al., Addison-Wesley,1995). This pattern allows the interface of an object to bridge to several different implementations.

Enhydra's XMLC compiler has built-in support for the Bridge pattern. Using it requires both the "-generate both" and "-implements" options on the XMLC compiler. To use this pattern effectively requires a little up-front analysis. What we need to look for is classifications or classes of pages. For example, we may find that our application has several HTML forms. Yes, each instance of the HTML form is different. It may have different input elements appearing in a unique order and position, but closer examination reveals that all the HTML forms are built from the same finite set of elements. This is a sure sign of a correct classification. I think you'll find only a small number of classes (form, detail display, summary display, etc.) in your applications.

When a classification is discovered, the next step is to create some code that defines the class (the HTML Form class). This definition is used as a template to create all instances of the class required by the application. Here the class is defined in HTML. The class HTML file must contain one example of each of the elements needed to create the instances. For example, our form class will require sample submit and reset buttons and several input element samples. Each sample element should have a unique ID attribute so the XMLC compiler will generate methods that make it easy to get hold of these elements when we start programming.

With the class defined, we're ready to generate the abstract interface and reference implementation Java classes. Your XMLC command will need to use the "-generate both" option (see Listing 1). This will cause XMLC to generate an abstract interface class (see Listing 2) that we'll use in our servlets. The implementation generated at this stage can be used as a reference implementation (see Listing 3) so you can start coding your servlets right away. This is great because developers can start building the application while the art department is still working their magic on the look and feel. *Note:* Files are generated by compiling the source code (on the Web at www.sys-con.com/xml/source.cfm), which requires ANT (see http://jakarta.apache.org.ant).

After this, you can add new looks to your application by creating a new class definition HTML file that has the same elements as the original. The elements themselves can be different, but every ID attribute in the original class must be in the subsequent classes. For example, the reference implementation HTML file might have a plain button element for a submit button and later implementations can define fancy rollover buttons. As long as the ID attribute matches the original class, it'll work just fine.

The question that always comes up is, "What if I miss one of the attributes?" XMLC's Bridge pattern support can help. Subsequent implementations are processed with the "-implements <interface>" option. If the abstract interface is used with this option, the generated Java implementation will implement the abstract interface; if any of the ID attributes don't match or are missing, you'll get a compile error. Because you have compile time check-

AUTHOR BIO

Nick Xidis is a senior principal architect with Iconixx in Overland Park, KS.

ing, you can safely let other developers produce HTML class definitions.

Abstract Factory Pattern

This pattern also comes from *Design Patterns*. The Abstract Factory creates instances of an object without the client actually knowing what the implementation class is (see Figure 2). We use the pattern because we don't want our servlets to know which implementation from the Bridge pattern we're using. That way the servlet is dependent only on the interface, not on any actual implementation.

To use this pattern we need to have our HTML classes defined because we'll need a create method for each of our interface classes on the Abstract Factory interface. Then we'll need to create a concrete factory implementation for each related group of classes. In our case each look and feel needs its own factory implementation. Coding all these factories was tedious, to say the least, but Java's introspection capabilities and XML can help. We created a factory implementation that reads in an XML configuration file (see Listing 4). The XML file defines which HTML class the factory implementation should instantiate. This means that we need only one concrete factory implementation that changes its behavior based on the XML file that it loads. How do we make sure we load the right XML file, and aren't there scalability limits to this approach? The answer to both questions is yes, of course, or I wouldn't have brought them up.

Factory Finder Pattern

This pattern deviates from the Abstract Factory pattern. In this case the objects created by the Abstract Factory are more Abstract Factories. A good example of this pattern is the CORBA Life Cycle Service. For our purposes we'll use a Factory Finder (see Listing 5) to get the correct factory that will create the HTML instances. The Factory Finder has a single method that returns the correct factory object based on supplying a string name. The name we used was the name of the user's financial institution–supplied parameter in the HTTP request.

We started out with an XML file–configured Factory Finder (see Listing 6) that used introspection to create instances of all the factories that were required. We found that when several thousand factories were loaded, the Factory Finder became difficult to maintain. The XML file got really big and the Factory Finder consumed a lot of memory, so we switched to an implementation of the Factory Finder that used a relational database for lookup and created an instance of factories on the fly. This too had scalability limits so we finally settled on an implementation that loaded data from our LDAP directory. This directory offers a much better read performance than a relational database and you can get some pretty cool administration tools to help with care and feeding.

Putting It All Together

Okay, we've got a Factory Finder that returns a Factory class that can create objects that are the correct HTML class for the branding we need. The HTML classes have examples of all the elements required to make the screens our application requires and an empty workspace identified by a "div" tag.

To build an application you need to take all of the above, add a servlet or two, and stir. Each servlet is passed the name of the look and feel to use. The servlet then uses the name to look up the correct Factory in the Factory Finder. The Factory Finder will find the Factory either in its own cache or the LDAP directory. The servlet now uses the Factory to get an instance of the HTML class. It then copies the example elements and puts them in the work space marked by the "div" tag.

Since all of the look and feel is encapsulated in the HTML class, the servlet has no idea what the page looks like – it's just putting the dynamic elements into the page wherever it finds the workspace. Finally, the example elements are removed from the HTML class and the class returns HTML to the browser. The servlet only knows the class interface and has no direct dependencies on anything to do with the look and feel.

In about 20 or 30 minutes, using these techniques, we can deploy a new look and feel that is consistent across all our applications. An added benefit is that we can give the template HTML file to a customer's Web designers and they can quickly create a new look and feel for the whole application. Since the template is validated by XMLC and the Java compiler as containing all the correct elements (e.g., it has to match the defined interface exactly), it's safer than handing out JSP pages for modification by third-party developers. Also, since XMLC supports HTML and XML, we can have the same code base serve up HTML as well as almost any flavor of XML. All we need is a different template.

NXIDIS @ ACM.ORG

Open Source Enhydra XMLC

Enhydra XMLC is a tool for creating Web and wireless presentations. As an open-source technology hosted at http://xmlc.enhydra.org, it's commonly used as an alternative to JavaServer Pages development because of its appeal to IT architects concerned about the long-term maintenance issues presented by JSP development.

XMLC's emphasis on precompiling HTML or any XML pages into "template objects" completely avoids the need to insert scripting or Java code into the markup page. Instead, Java logic manipulates the page by operating on the template object. By representing markup pages as a W3C standard DOM template class, XMLC enforces good programming practices since developers can never "break the rules" and resort to embedding Java in the markup.

This appeals to IT architects concerned about developers who are using JSP to intermingle markup and Java code, setting the stage for future maintenance nightmares. It also gives architects a Java path to XML development without requiring development (and support) of yet another language, such as XSLT.

During development of the presentation page, designers and/or developers can identify areas of dynamic content by inserted ID attributes inside markup elements. These attributes are later used by the XMLC compiler to create access methods that can be used by the developer to manipulate the page with new information.

Additional benefits of XMLC programming give designers the ability to leave mocked-up content in the page, creating a true WYSIWYG presentation of the page before the presentation is rendered during execution. This is impossible with JSP. XMLC also supports the ability to detect changes made to the original document page. When a change is detected, the XMLC environment can be configured to auto-recompile and reload the page, thus making it possible for designers to make presentation changes without the required involvement of the Java developer.

XMLC has evolved since its introduction by Lutris in early 1999 to embrace WML, XHTML, cHTML (i-mode), and VoiceXML, thanks in large part to the contributions made by Enhydra.org members from around the world. Barracuda, another open-source effort, is a presentation framework that takes advantage of XMLC to add component-based GUI development modeled after the Swing libraries of the Java Foundation Class.

XMLC is part of the larger Enhydra open-source project, founded and stewarded by application server vendor Lutris Technologies. Enhydra has gained notoriety by virtue of its focus on open-source application server technologies, including, most recently, EnhydraME. EnhydraME was launched last summer to extend Web service technologies such as SOAP (kSOAP), XML parsing (kXML), and UDDI (kUDDI) to devices turning. The result is that J2ME devices from Motorola, RIM, and NTT DoCoMo can use EnhydraME technologies to become microapplication servers.

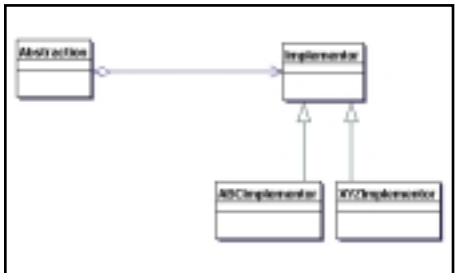


FIGURE 1 Bridge pattern

FIGURE 2 Abstract Factory pattern

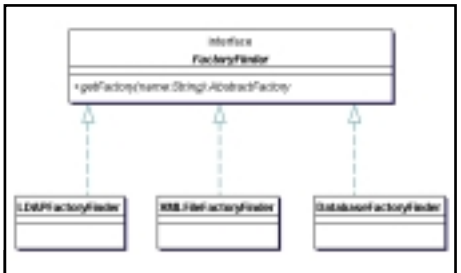


FIGURE 3 Factory Finder pattern

wireless EDGE conference&expo

MAY 7–9, 2002

Santa Clara Convention Center
Santa Clara, California

STRATEGIES & SOLUTIONS FOR UNWIRING THE ENTERPRISE

International Wireless Business & Technology Conference & Expo

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

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

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.

CONFERENCE TRACKS

TRACK 1: DEVELOPMENT

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

TRACK 2: CONNECTIVITY

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

TRACK 3: WIRELESS APPS

- Education
- Healthcare
- Entertainment
- Transport
- Financial Services
- Supply Chain Management

TRACK 4: HARDWARE

- Cell Phones/WorldPhones
- PDAs
- Headphones / Keyboards / Peripherals
- Transmitters / Base stations
- Tablets

TRACK 5: MANAGEMENT

- Wireless in Vertical Industries
- The WWWW
- 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

Arie Mazur
President, Chief Executive Officer, and Chairman, Slangsoft

Chris Bennett
Architect, Freedom Technologies

Daniel Elliott
Senior Vice President, Mobile Business, CompuCom

Douglas Lamont
Professor of Marketing, DePaul University

Kenneth Leung
Business Development Manager, Retail and Emerging Technologies, IBM

Keith McIntyre
Vice President and Chief Technologist, Stellcom

Steve Milroy
Wireless Technologist, Immedient

Tony Wasserman
Director, Mobile Middleware Lab, Hewlett-Packard Company

WWW.SYS-CON.COM/WIRELESSEDGE2002

Exclusive Sponsorship Available

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

SYS-CON MEDIA

ONLINE REGISTRATION NOW OPEN

WWW.SYS-CON.COM/WIRELESSEDGE2002

Or, To Register By Phone
Call: (201) 802-3069

	Development	Connectivity	Wireless Applications	Hardware	Management
			MAY 8, 2002		
8:30-9:45	Embedded Java (Bill Ray, Network 23 Limited)	A Real Time Model of (3G) UMTS Access Stratum (Niloy Mukherjee, MIT Media Laboratory)	Using Mobility to Streamline Traditional Business Processes (Dan Elliott, CompuCom)	Java Software Performance On Wireless Devices: Myths and Realities (Ron Stein, Nazomi)	LMDS - The Last Mile Enabler of Class 5 Revenues with VoIP (Ed Peters, Ensemble Communications, Inc.)
10:00-11:15	**KEYNOTE**				
11:30 A.M.	**EXPO FLOOR OPEN**				
	LUNCH BREAK				
1:30-2:30	XML & Wireless Technologies (Karl Best,OASIS)	Securing Wireless Data Via Smart Card (Joseph Smith, New Dominion Software)	Collaboration for Wireless Warriors (Timothy Butler, SiteScape, Inc)	User Interactivity for Information Appliances (Arie Mazur, Slangsoft)	Leveraging Wireless In Customer Acquisition and Retention (Kenneth Leung, IBM)
2:45-3:45	**KEYNOTE**				
3:45-4:45	Developing Mobile Web Applications (Tony Wasserman, Hewlett-Packard Company)	In-building Wireless, the Next Frontier (Mary Jesse, RadioFrame Networks)	Turbocharge Mobile Applications with J2EE (Dr. Jeff Capone, Aligo, Inc.)	Cell Phones/ WorldPhones (Speaker TBA)	Mobilize Your Enterprise (Chris Bennett, Freedom Technologies)
5:00-6:00	VoiceXML Workshop (Bryan Michael, BeVocal)	Satellite Broadband (Speaker TBA)	Mobile Portals - The First Step Towards the Mobile Enterprise (Tony Wasserman, Hewlett-Packard Company)	PDAs (Speaker TBA)	Wireless Solution Return On Investment (ROI) In the Real World (Steve Milroy, Immedient)
			MAY 9, 2002		
8:30-9:45	Brew, I-Mode, WAP, and J2ME: How the Battlefield Is Shaping Up at the Start of the Mobile War (Reza B'Far, eBuilt, Inc.)	The Future of IP Mobility (Antti Eravaara, NetSeal, Inc.)	Multimodality: Revolutionizing Our Wireless Lifestyles (Arvind Rao, OnMobile Systems)	Tablets (Speaker TBA)	Marketing Value in Automotive Telematics Through Mobile Communications (Douglas Lamont, DePaul University)
10:00-11:15	**KEYNOTE**				
11:30 A.M.	Guide to Developing Applications with Micro Java (Kurt Baker, Kada Systems)	**EXPO FLOOR OPEN**			
	LUNCH BREAK				
1:45-3:00	Building Secure Mobile Solutions (Keith McIntyre, Stellcom)	Smart Card Communications (Bill Ray, Network 23 Limited)	Developing New Applications Using VoiceXML (Jonathan Taylor, Voxeo)	PDAs (Speaker TBA)	Policies and Profiles: The Key to Mobile Data Services (Doug Somers, Bridgewater Systems)
3:15-4:30	Bluetooth™ Wireless Technology and Java ™ Technology (Michael Portwood, Exuberance, LLC)	UMTS/3G Networks (Speaker TBA)	Instant Messaging and Wireless Computing Collide (JP Morgenthal, IKimbo)	Transmitters / Base Stations (Speaker TBA)	Total Business Solutions B2E (Speaker TBA)
4:45-6:00	Creating Carrier Optimized Wireless Internet Applications (David Young & Victor Brilon, Lutris)	Wireless LANs/Bluetooth (Speaker TBA)	Rapid Development of Successful Wireless Applications (Rod Montrose, AVIDWireless)	Headphones / Keyboards / Peripherals (Speaker TBA)	Wireless Sales & Marketing (Speaker TBA)

REGISTER BEFORE March 1 And Save Up To $400

Plan to Attend the 3-DAY Conference

PRODUCED BY
SYS-CON EVENTS

May 7 will feature full-day tutorials. Consult www.sys-con.com/wirelessedge2002 for up to-the-minute conference news.

PROGRAM SUBJECT TO CHANGE

Getting a Jump into Web Services

Once upon a time, the only technologies you needed to know to develop Web applications were HTML and Perl. Nowadays, while HTML is still around, the days of using Perl for development of sophisticated Web-based applications are long gone. Sure, it can still be done, but there's a whole can of alphabet soup full of acronyms for connecting Web-based services that is the next big thing. It's those acronyms, and the connection of the services they define, that this book covers.

Checking in at just under 600 pages and nine chapters, the book's seven authors hit many of the latest buzzwords straight out in the title: Web services, Java, XML, SOAP, WSDL, and UDDI. Targeting the accomplished professional who knows about Web and distributed applications, I found the book to start a little simpler than expected. Perhaps it's just me, but if you're going after the professional developer as opposed to the newbie, you have to assume a certain level of knowledge, and in this case you've read 20% of the book by the time you get past the overview and XML primer.

The book's explanation of Web services explores the necessary set of architectures. Starting off with SOAP, and not the Ivory kind, the book introduces Web services development by continually building up the Web site for a fictitious company called SkatesTown. The continuity this provides between examples is a nice feature when compared with books that offer many disconnected examples.

Moving beyond SOAP, the book travels next to the land of Web services creation. Here you'll learn about using the open source Apache Axis architecture, where *Axis* stands for *Apache Extensible Interaction System*. The explanation is well done, and provides a deep level of coverage from someone who at least appears to know what's going on. There's even some helpful configuration pointers, and of course the necessary deployment information. With Axis still in an early alpha release, at a minimum you'll find this chapter helpful, but the specifics of the capabilities are subject to change.

The next topic is the use of SOAP for e-business. This is less an API review–type chapter and more oriented toward exploring system-wide issues that need to be dealt with in real-world systems. Most important is the discussion of security, though there is also coverage of such topics as high availability and application integration. The overall content here is probably appropriate for the level of the book, but so much more can be said on the different topics here.

Up next are the descriptions of Web services. This is where items come up such as the Web Services Definition Language, how you publish services for others to discover and use, and the Universal Description Discovery and Integration standard for how to discover services. There's plenty of syntax coverage of WSDL for describing services and of UDDI for maintaining registries of services. You'll definitely learn what role these two components play in the big scheme of things with Web services and how to use them for your own systems.

The final two chapters before the glossary deal with interoperability and futures. Since as a whole Web services aren't language-centric, the authors explore other environments where Web services, even Perl, can be used. There's a nice little .NET/C# example, too.

The futures chapter isn't about the future of the different alphabet soup acronyms. That future is covered in the individual chapters on each acronym. Instead, the authors present a vision of the future of the Web, including a look at the semantic Web, peer-to-peer computing, and grid computing.

Without belittling the book's accomplishments, it does have some faults. I found the inline icon indicating that a word was found in the glossary to be very annoying. It broke up the natural reading rhythm, especially in the earlier chapters where there were frequent glossary entries. I also found the expansion of the acronyms *JSP* and *EJB* consistently wrong. JSP is *not* Java Server Pages and EJB is *not* Enterprise Java Beans. Sidebars following seven pages after they were mentioned in text were also awkward.

One useful feature throughout the book is the end-of-chapter resource lists. They're full of links to sites on the different technologies. Some of these technologies are evolving, and the links point you in the direction of the latest news and information. Too bad these aren't available online, though.

Overall, the book provides a great start toward developing Web services. The content is easily understood and would seem to hit the mark for the right audience. Unfortunately, after reading the introduction and examining the level indicated on the cover (Professional), my impression was that the level would be higher, and that just isn't the case.

[Reviewed By John Zukowski]

Author Bio

John Zukowski conducts strategic Java consulting with JZ Ventures, Inc., and serves as the chief evangelist for Spidertop. His latest books are Java Collections *and* Definitive Guide to Swing for Java 2 *(2nd edition) from Apress.*

JAZ@ZUKOWSKI.NET

Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI
by Steve Graham, Simeon Simeonov, Toufic Boubez, Glen Daniels, Doug Davis, Yuichi Nakamura, Ryo Neyama

Published by Sams
Pages: 600
List Price: $49.99

XML-J ADVERTISERINDEX

ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com		68
BEA eWorld	www.bea.com/events/eworld/2002	404-240-5506	13
CTIA Wireless 2002	www.ctiashow.com		45
DataMirror	www.datamirror.com/resourcecenter	800-362-5955	9
HitSoftware	www.hitsw.com3	800-345-4001	6
Java Developer's Journal	www.sys-con.com	800-513-7111	51
JavaOne	http://java.sun.com/javaone	888-886-8769	39
JDJ Store	www.jdjstore.com	888-303-JAVA	23, 62, 63
Macromedia	www.macromedia.com/downloads	888-939-2545	4
.Net Developer Conference & Expo	http://events.internet.com		19
Sonic Software	www.sonicsoftware.com		2
SYS-CON Media	www.sys-con.com	800-513-7111	27, 47, 61
SYS-CON Reprints	www.sys-con.com	201-802-3026	31
VoiceXML Planet Conference & Expo	http://events.internet.com		19
Web Services Conference & Expo	www.sys-con.com	201-802-3069	40, 41
Web Services Journal	www.sys-con.com	800-513-7111	54
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	800-513-7111	25
WebSphere Developer's Journal	www.sys-con.com/websphere	800-513-7111	31
Wireless Business & Technology	www.sys-con.com/wireless	800-513-7111	53
XML Global	www.xmlglobaltechnologies.com/newangle	800-201-1848	67
XML-Journal	www.sys-con.com	800-513-7111	49, 61

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.

Next Month...

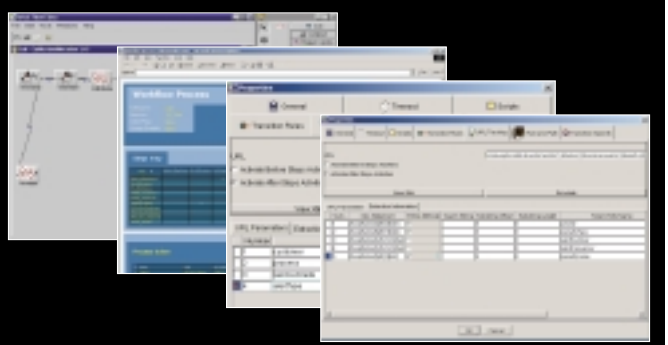
XML Glue
Using an XML workflow and integration layer for Telecommunication Providers
by Ron Ben-Natan

XML Messaging
Use JAXM to exchange SOAP messages
PART 1 of 2
by Mike Jasnowski

A Client for Testing Server Side XML Applications
How to speed up the process of building and debugging server-based XML applications
by Jon Strande

ANT: An Introduction by Example
A platform independent alternative to shell script-based build tools
by Ann Black

XML Is a Treasure My Heart Cannot Deny
The magic of an XML stylesheet
by Tod Emko



XML JOURNAL

Don't miss the March issue!

JDJSTORE.COM GUARANTEED LOWEST PRICES!

Guaranteed Best Prices

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

Terms of offer:

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

Prices subject to change.
Not responsible for typographical errors.

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

BORLAND
JBuilder Enterprise V6.0
FREE with order...Your choice of one of "The Complete Works" CDs
Programmer's Paradise Price - $2899 (1/8/02)

JDJStore.com **$2749.99**

ONE FREE CD
with order of either
JBuilder Enterprise V6.0 or Visual Café Enterprise Suite

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

WEBGAIN
Visual Café Enterprise Suite
FREE with order...Your choice of one of "The Complete Works" CDs
Programmer's Paradise Price - $2995 (1/8/02)

JDJStore.com **$2749.99**

MACROMEDIA
ColdFusion Server 5 Professional Edition

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

ColdFusion Server 5 . . . . . **$1219.99**

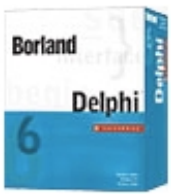
SYS-CON MEDIA
XML-Journal: The Complete Library

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

XML-Journal: The Complete Library **$53.99**

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 **$2849.99**

INSTALLSHIELD
InstallShield Professional 6.3

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

InstallShield Professional 6.3 . . . . . **$999.99**

INSTALLSHIELD
InstallShield Professional 6.3 Upgrade from 6.X

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

InstallShield Professional 6.3 Upgrade from 6.X **$499.99**

eHELP
RoboHELP Office 2002

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

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

RoboHELP Office 2002 . . . . . **$898.99**

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 . . . . . $348.99

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 . . . . . $2945.00

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. $169.99

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 . . . . . $3540.00

METROWERKS

CodeWarrior Pro V6.0

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

CodeWarrior Pro V6.0 . . . . . $349.99

WEBGAIN

VisualCafé Expert 4.5.1

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

WebGain VisualCafé Expert Edition. $899.99

ALTOVA

XML Spy v 4.2 Suite

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

XML Spy v 4.2 Suite $399.99

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 handwrite 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 . . . $198.99

MACROMEDIA®

JRun™ Server 3.0/3.1 Enterprise 1 CPU Licenses

JRun™ 3.0/3.1 is an easy-to-use J2EE application server and integrated development environment for building and deploying server-side Java applications. From e-commerce to business automation, JRun is the easiest way for developers to deliver advanced business systems faster and at a lower cost than you'd ever thought possible.

JRun Server 3.0/3.1 Enterprise (1 CPU Licenses) $4499.99

HP

Jornada 547 Handheld Device

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

Jornada 547 handheld Device. $299.99

INSTALLSHIELD

DemoShield 7.0 Full

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

DemoShield 7.0 Full $599.99

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 $1234.00

XML News

Updated DocExpress Adds XML Links

(Irvine, CA) – Telelogic has released a new version of Telelogic DocExpress, a leading automated solution for engineering documents. DocExpress 3.0.1 provides robust support for XML data files and tightens integration with other Telelogic tools and a wide range of third-party products.

Key enhancements include improved support for Telelogic DOORS, enhanced support for Adobe FrameMaker, a link to Telelogic CM Synergy, and support for Microsoft Word XP.
www.telelogic.com

Schema-Based Spec for Business Reporting Finalized

(New York, NY) – The XBRL.org Consortium has finalized its enhanced XML schema-based specification for business reporting, which can now be used by companies and organizations to put financial statements and other reporting information into XBRL (eXtensible Business Reporting Language).

In other news, Bank of America, one of the world's largest financial service companies, is piloting the use of XBRL to collect data from customers that are required to provide financial statements on a regular basis for lending and credit analysis purposes.

Members of the XBRL.org Committee represent the largest financial, accounting, and technology organizations around the world.
www.xbrl.org

BusinessObjects Developer Suite 5.5 Unveiled

(San Jose, CA) – Business Objects has updated BusinessObjects Developer Suite, its development toolkit for customizing, integrating, and extending the Business Objects BI platform. The suite includes the standard development environments of JavaServer Pages (JSP), Active Server Pages (ASP), and Microsoft Visual Basic for Applications (VBA), as well as an extensive object model, the Business Objects standard API, and a wide assortment of sample programs and documentation.
www.businessobjects.com

B-Bop Simplifies MS Word to XML Publishing

(Burlingame, CA) – B-Bop Associates, Inc., has released Xfinity Author wX 2.1, the latest version of its Microsoft Word-to-XML authoring software. The software offers new functionality to further simplify the creation of structured documents in XML directly from the Microsoft Word environment.

The new release allows users to save their content to any XML industry standard or user-defined DTD, such as RiXML, NewsML, HR-XML, DocBook, XBRL, or even HTML. It also provides comprehensive support for international languages via support for both ISO-8859-1and UTF-16 character sets.
www.b-bop.com

New Breeze XML Binder 3.0 for Java

(Austin, TX) – Breeze Factor's latest data-binding product, Breeze XML Binder version 3.0 for Java, has a number of new features, including extensive support for the W3C XML Schema Recommendation, W3C namespace support, W3C DOM interoperability, enhanced marshaling/unmarshaling performance, more code-generation control, and automated compilation.

Breeze XML Binder is available via download directly from The Breeze Factor's Web site.
www.breezefactor.com

OASIS Invites Submissions for Test Suite

(Boston) – OASIS has issued an open invitation for public input on the creation of its XSLT/XPath Conformance Test Suite.

Organizations and individuals that have created test files for XSLT processors are encouraged to submit their work to the OASIS XSLT/XPath Conformance Technical Committee, which plans to corroborate submissions and publish an amalgam test suite for public use. Formal submissions from both IBM/Lotus and Microsoft have already been received.

To conform to the OASIS framework, submitters must catalog their test file collections according to OASIS published document models, and they must employ available validation processes that utilize XSLT and XPath. Complete submission guidelines are posted on www.oasis-open.org/committees/xslt/.

MicroBilt Introduces Interface for Obtaining Credit Data

(Kennesaw, GA) – MicroBilt, a leading provider of credit access and reporting, has launched an XML interface for their software developer's kit and added new functionality to Credit-Commander.com.

Credit professionals can be up and running in days with MicroBilt's XML interface, giving them access to the credit

Sun Introduces Java XML Pack

(Santa Clara, CA) – Sun Microsystems has announced the immediate availability of Java XML Pack, an easy way for developers to get all the latest XML and Web services technologies for the Java platform.

The download bundles key industry XML and Web services standards, giving Java developers the technologies they need to quickly and efficiently build Web services. The Java XML Pack is available from Sun's Web site at http://java.sun.com/xml/downloads/javaxmlpack.html.

Software AG, Altova Enter Strategic Partnership

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

The complementary products give XML Spy users easy and low-cost access to Tamino, enabling them to natively store and process XML data from new applications. In turn, the partnership extends Software AG's reach to Altova's community of 400,000 developers.
www.xmlspy.com
www.softwareagusa.com

XML News

report and each of the data elements behind the report.

In addition, MicroBilt has released their flagship Decision Table product on their Web site at www.CreditCommander.com. Previously available only through their SDK, Decision Table offers credit managers the ability to automate their credit process with an easy-to-use Web interface.
www.microbilt.com

Pageflex Releases Persona 1.5

(Cambridge, MA) – Pageflex, Inc., is shipping Pageflex Persona 1.5, a desktop application for variable content publishing. Persona 1.5 provides improved optimization for PPML (personalized print markup language) and CreoScitex VPS (variable print specification) output drivers. In addition, Persona supports other vendor-specific output drivers such as Indigo JLYT and Xeikon Intellicache.

Persona is the first variable publishing solution to use XML as the intermediate data format between a profile database and the page composition process.
www.pageflexinc.com

ActionPoint Updates InputAccel Capture Software

(San Jose, CA) – ActionPoint, Inc., is now shipping its flagship product, InputAccel Version 4.0, incorporating the latest capture technologies.

Several key features of the release are color utilization, multistream capability, ability to output information using XML, two new PDF solutions, and support for the latest enterprise content management systems.
www.actionpoint.com

OTA Releases XML Spec for Public Review

(Alexandria, VA) – The OpenTravel Alliance (OTA) has released version 2001C of its XML specification for public review.

Version 2001C includes both the ebXML-OTA mapping and best practices documents, and enables users to request and respond to airline flight availability; request and book comprehensive package holiday tours along with a variety of itinerary elements such as transportation, accommodation, and car rental; check on the availability of golf courses and book a tee time; search for specific hotels, check for availability, and make a reservation; check the availability of car rentals and simplify the steps to making that reservation; and get and book quotes on travel insurance.

To view and comment on 2001C, visit www.opentravel.org/opentravel/2001c.cfm.

i:FAO Debuts cytric v7

(Dearborn, MI) – i:FAO is starting installation and deployment of cytric v7, its XML-based Internet software for booking and management of business travel.

Written entirely in Java, cytric v7 is the first business travel e-procurement solution that combines all the benefits of online applications with an open, enterprise-class infrastructure.
www.ifao.net

Sueltz Elected to Amgen's Board of Directors

(Thousand Oaks, CA) – Amgen has elected Patricia Sueltz, executive vice president and general manager of Sun Microsystems' Software Systems Group, to its board of directors.

At Sun, Sueltz is responsible for all aspects of the company's software systems business, including Java/XML, Solaris, and the Sun Developer Network.

Amgen is a global biotechnology company that discovers, develops, manufactures, and markets important human therapeutics based on advances in cellular and molecular biology.
www.amgen.com

Enhanced SDK, Soap/XML-Based Interfaces for PRISM

(Honolulu) – Inovaware Corporation, a leading developer of billing and customer care software for the next-generation Internet services and communication industry, has released an enhanced software development kit and new interfaces for its PRISM platform.

The new release for usage import permits customers and systems integrators to rapidly create usage import modules on their own. Primary benefits include fast time-to-market for innovative pricing models and self-sufficiency as SDK-created modules effectively behave as an out-of-the-box PRISM module.
www.inovaware.com

Vordel Offers Support for Intel Itanium Processor

(Dublin / Santa Clara, CA) – Vordel has announced the interoperability of its signature product, TalkXML, and Intel's Itanium processor.

Vordel TalkXML provides a digitally signed audit trail for commercial transactions. The product is designed for the data conversion and routing requirements that are characteristic of complex financial trading environments. TalkXML's compatibility with Intel's Itanium 64-bit processor allows the software to process large volumes of data at high speed.
www.intel.com
www.vordel.com

Nuance Ships Nuance 8.0

(Menlo Park, CA) – Nuance has unveiled a new version of its flagship speech recognition and natural language understanding server software.

Nuance 8.0 provides accuracy improvements for calls placed from wireless and noisy environments, support for the VoiceXML 2.0 standard, and Linux support. It also features Dynamic Language Detection, which allows voice applications to automatically detect in real time the language the caller is speaking. The software is available in 26 languages.
www.nuance.com

NeoCore Names New President and CEO

(Colorado Springs, CO) – NeoCore has chosen Eric "Ric" Miles as its new president and chief executive officer. Miles most recently was the executive vice president of worldwide operations at FrontRange Solutions, also based in Colorado Springs. His previous experience includes executive and managerial positions at PeopleSoft, Vantive, Compaq Computers, and Tandem.

Company chairman and founding CEO Tim Dix stated that Miles's hiring is the culmination of a more than yearlong search to find the right industry-based person to lead NeoCore's operations as the company grows and pursues major national and international business opportunities.

Dix, a cofounder of NeoCore, will continue to work full time for the company as senior consultant and its chairman.
www.neocore.com

I Am the Official Mascot of XSLT

Noncompliancy has got to go

The Transforming Robot is a 20-ft-tall humanoid machine that can turn into an airplane. He used this talent to become a film and television star in the 1980s, but he soon found himself without a job in the '90s. He was excited to hear about XSLT, remarking at the code, "There's more here than meets the eye!" And then he hunched over and groaned sadly.

BY **A TRANSFORMING ROBOT LOOKING FOR WORK**

Hey, I was pretty cool for a while. You all watched me on TV and film for quite a few years. And I think my day has come again. XSL Transformations have arrived. As a high-profile transformer of sorts, I declare myself the official mascot of XSLT.

And, honestly, what choice do you have? I'm huge, and I advocate transforming. I mean, just who else does that? Would you choose a monster like Mr. Hyde or the werewolf over me? I fought for justice, and for proper transformations within a specified set of rules. With those monsters as your mascots, you'd just have a bunch of programmers eaten by a dog boy, or a sociopath running about your offices! Is that what you want?

Well, I shouldn't assume too much. You did choose to watch other, more pathetic aliens like ALF over me. How am I to know that you wouldn't choose a recycled carpet with a snout over me again this time? It would take mature, discerning programmers and managers to choose me this time. And, er, ones that still like cartoons about giant robots.

Now, just because you discarded me, turning off my TV shows and choosing a bunch of mutant turtles over me, that doesn't mean I'm still bitter. That was a long time ago. A long, long, depressingly long time ago. But just to be clear, we're not square. If you accept me as the new mascot, I demand a few changes be made. First, all non-W3C-compliant XML scripts will be renamed Evil Decepta-bot Scum. Second, all programmers will make a "wecka-wacku" sound when a document successfully transforms. Third, all of you will fear me. Oh, wait, did I say that out loud?

And, uh, if you aren't convinced yet, think about a few more things. Think about the fact that everyone thinks of me when they hear "transformation." They don't think about cascades, or styles of any sort. They think of me, kicking butt. Wouldn't you want everyone to know how cool your stylesheets are, with a smoking laser gun and me as your symbols? I've even learned to make a cool "X" with my arms every time I destroy an evil Decepta-bot with my gun of unlimited compliance. Yeah, I've thought about this. I've got ideas about it. Why are you looking at me like that?

All right, if you're going to be close-minded, allow me to expand on some of my ideas. How about this? A picture you can put on your XML products: me holding some of your validating software, exorcising an unruly stylesheet. No? Well, I think it's better than me continually having to use this giant laser gun to make a living. I'm trying to stop the violence! Doesn't that count for anything? All the rest of my friends are making a living in violent Japanimation movies. They want your children to grow up filled with hate for each other. You can trust me, though; I've reformed. I've renounced my murderous ways.

You wouldn't want that choice to go unrewarded, would you? I'd much rather hold up a cup of coffee and smile, symbolizing the partnership between Java and XML, than hold up the heads of Un-compliant-Bot and evil Unclosed-Tag-a-Bot, my mortal enemies. I could show off my ability to perform monologs, expounding on how I'll take revenge on evil boss bots for killing my leader.

I want to move on from that, though it *was* my specialty. Boy, I'd bring the house down when I shook my fist at the sky, standing in the junk pile that used to be my best friend. But I won't look back.

So, I've put forth my plea, my logic, my last shot at revitalizing my career. Now, come on, if an unholy demon of Satan can somehow become the mascot for a Unix operating system, how can you turn me away? If you don't take me, my only other choice is to sell myself on eBay. You'll be sorry when I'm a collector's item. Last chance! Wait, where are you going?

WRITTEN BY **TOD EMKO**

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

TEMKO@TOASTEDPIXEL.COM

XML Global Technologies

www.xmlglobal.com/newangle

Altova

www.altova.com