XML & JAVA: THE HOW AND THE WHY

Volume: 4 Issue: 9, September 1999

JAVA DEVELOPER'S JOURNAL

The World's Leading Java Resource

XML FOCUS

SYS-CON PUBLICATIONS

SEAN RHODY, EDITOR-IN-CHIEF

Two to Tango

Sometimes I think no one reads the editorial. Normally I receive maybe no more than fifty comments concerning any editorial. I kid myself that I do a good enough job lining up the content for the magazine that no one has any complaints, and that they post each month's editorial on a wall for all to see. In reality, I hope it doesn't show up on that many dart boards – I know programmers.

Our JavaOne issue ***(JDJ*** Vol. 4, issue 6) wasn't one of those times. I've been writing about programming for over five years, and I've got to say that the response to my "XML Mambo" column was nothing short of a landslide. I couldn't have selected a more charged topic to write about, a fact I'm very pleased with.

To recap briefly: my column questioned XML as the next "killer app." While I feel XML is important, it's my thought that it's overhyped. In my mind that was the gist of the article.

Now I'm either a genius or a schmuck.

The e-mails came fast and furious: those who thought I was so right – XML is just marketing hype – and those who thought I was a blithering idiot – hadn't I ever tried to code server pages in HTML? (Fact is, no, I haven't. That sounds particularly ill-advised in my opinion.) But from an editorial standpoint I had to stand up and take notice: XML is important to you. Maybe it's the next killer app…maybe it isn't. Clearly though, you were concerned about the impact XML would have on the world, particularly the Java community. And so was I.

In fact, we here (here being a virtual kind of thing: I'm in New Jersey, Alan's in Scotland, Ajit's in Texas) at **SYS-CON** have been having numerous discussions concerning XML. So many that I lose the thread of the topics regularly. One thing we all agree on though is that XML is something we need to cover closely.

Hence this issue. Welcome to our special focus issue on XML. On subsequent pages you'll find numerous articles relating to XML, and to Java. We're exploring the topic, possibly in preparation for bigger things. (Can you say *XML Developer's Journal*? I knew you could.) Because we know that it's important to you. Either from a career possibility standpoint, or at least for the impact it will have on your lives as Java programmers. Inside, you'll get more information about XML, as well as the opinions of our sharpest minds here at the magazine.

I'd like to try to set some things straight, though. One, I don't think XML is pointless. To me it's objects without the methods – which sounds like a step backward, but may not be, as not every object truly needs methods. Second, I think it will have a big impact on e-commerce and EDI, as it's a simple, powerful way to agree on data transfers between organizations. It may even be a decent way to allow larger companies to interact with Web sites without having to have a human being sitting in front of a browser.

But I still stick to my guns when I say that XML is not the next killer app in the way that Java was. XML won't stop me from programming in Java the way Java stopped me from programming in PowerBuilder and, for others, VB or C++. It's not going to change people's minds about platform-independent coding or free them from the Tyranny of Redmont (I'm not sure Java did either, this is editorial license). Instead, I see XML as the SQL of the next century. SQL is everywhere, but few people point to it as a revolutionary product. XML will be like that. We'll adopt it as a good idea that should have been there in the first place, but it won't truly change our lives as developers.

And that's my opinion. As always, I welcome your comments and will try to respond to them. I hope you find this issue intriguing, insightful and useful.

sean@sys-con.com

AUTHOR BIO

Sean Rhody is the editor-in-chief of Java Developer's Journal. He is also a principal consultant with Computer Sciences Corporation, where he specializes in application architecture – particularly distributed systems.

WRITTEN BY AJIT SAGAR

Welcome to XML!

I have to agree with ***JDJ***'s editor-in-chief, Sean Rhody. The word *XML* seems to spark technological fires. The JavaOne Conference issue of ***JDJ*** (Vol. 4, issue 6) featured three articles on XML. Having written one of them, I share the experience of the flood of e-mails regarding this obviously hot and controversial topic. Aside from the folks who actually read what I write, others who have little clue about programming have asked me what XML is. As Sean mentioned, at ***JDJ*** we've been tossing e-mails back and forth about XML, where it is today, what it may mean and what role **SYS-CON Publications** should play in its evolution.

Admittedly, a lot of market hype surrounds XML. But that's true of any emerging technology. Four years ago people's opinions of Java ranged from a language that allowed you to do cute things on the Web to a platform that would revolutionize distributed computing. XML is in its infancy stages now. However, it holds the promise of unifying data formatting and display in computing as well as fulfilling the dream of real cross-platform data exchange. Mind you, this is just a promise. What it can do, and what it won't do, will be clearer over the next few years. Regardless, almost anyone who's doing anything in the world of e-commerce is taking a serious look at XML.

Before I ramble on, I guess I should introduce myself. I've been one of ***JDJ***'s contributing editors for more than two years now. I started with the **Cosmic Cup** column on the Java platform. This year my **e-Java** column covers Java and e-commerce. The purpose of this focus issue is to bring you comprehensive coverage of XML in all its aspects – good *and* bad, because there's a lot of confusion about this technology in the marketplace today. Our writers, who are experienced in various facets of the computing industry, will bring you the best coverage that can be found on this topic. This issue serves as a sample of the kind of information we intend to provide, possibly as an independent XML publication.

The articles in this issue aren't all "gung ho" XML. Israel Hilerio discusses the XML-Java relationship. He explains how XML relates to Java and why it's important for Java-based enterprise solutions, and dives into how the Java's dynamic class loading works with XML. Kang Lu covers EDI and XML. He brings a down-to-earth feel to the hype that surrounds XML and what it's really good for when complementing EDI. He relates this to the real world of human-to-human communications. Michael Boyd discusses RDBMS vis-à-vis XML and describes XML support in Oracle8*i*. In the spirit of the current issue, I discuss the XML-based data interchange format for the Online Ticket Store series of articles. Jason Westra explains how the newly released XML DTD for EJBs is becoming a standard for EJB deployment descriptors, another example of the synergy between the worlds of Java and XML.

This issue also features columns that focus on XML technologies. In **XML Corner** Andrew Watson talks about XML's role in middleware, specifically with reference to CORBA. Andrew explains how XML isn't a silver bullet, but can serve a complementary role alongside CORBA in middleware solutions. In this month's **e-Java** I attempt to describe the various product categories that exist in the XML market today.

In this issue we also offer ***JDJ***'s interview with Bob Sutor, the chief strategy officer of OASIS. Bob told us about OASIS and IBM's strategy regarding XML. In addition, Israel Hilario and Tija Ragas offer their reviews of two recently released XML books.

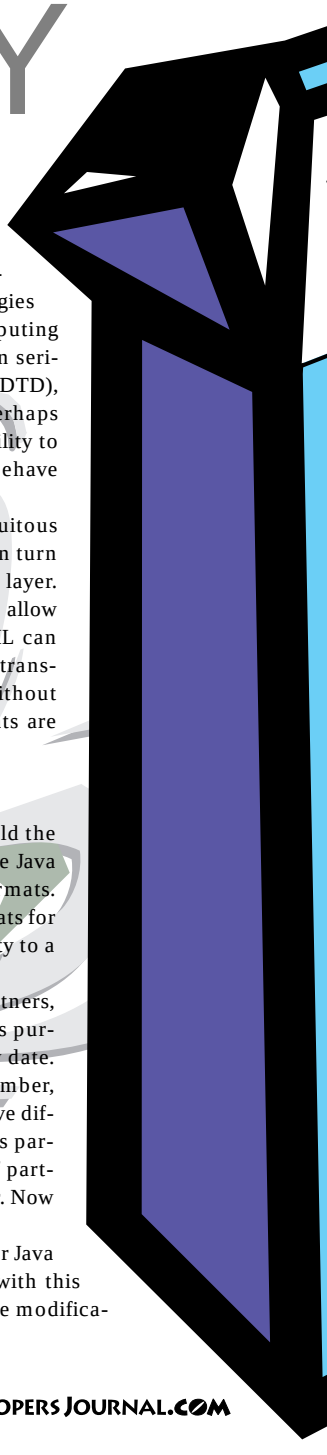
XML isn't an application-building technology. It's an enabler. Currently, XML is playing various roles in distributed computing, the primary ones being data format definitions and data display. Some purists look at it as a threat to existing technologies. XML fascists think it will solve world hunger. In my opinion there's a middle ground. I hope here at **SYS-CON Publications** we can help you identify how you can apply XML in your application domain to solve real-world problems. We look forward to hearing from you.

Ajit_Sagar@i2.com

XML & THE WHY

WRITTEN BY ISRAEL HILERIO

XML is to Java as cream is to coffee



Today the technical media talks a great deal about the Java platform and its importance in creating a ubiquitous Internet execution environment. While most of us have bought into this concept, other technologies that are emerging rapidly promise to smooth out the road to the computing promised land. XML is one of these technologies that needs to be taken seriously. There are many aspects of XML: Document Type Definitions (DTD), Style Sheets (XSL), Viewers, parsers, HTML 4.0 and data. Out of these, perhaps the most promising aspect of XML is its ability to represent data. Its ability to describe its document content via its markup mechanism allows it to behave like a universal data format for any number of applications.

Data representation using XML is a major step toward creating a ubiquitous data environment. XML allows authors to define their own tags, which in turn describe their content and make it possible to define a reusable data layer. Authors are able to leverage their document structure and meaning to allow the processing of special instructions on parts of their document. XML can also be used by two or more entities as a particular exchange format for transaction protocols. This allows XML documents to be manipulated without human interaction in batch mode. Some examples of exchange formats are defined by the Rossetta Net and Microsoft's BizTalk standards.

Why XML?

XML by itself has nothing to do with Java and vice versa. So why should the Java community care about XML? The answer lies in the data layer. The Java language alone doesn't provide a mechanism for standardizing data formats. Java programs need to rely on predefined, nonflexible, hard-coded formats for reading information. This makes it difficult to extend or add functionality to a program without breaking the existing code base.

Take a business scenario. Imagine that you do business with two partners, one on the West Coast, the other on the East Coast. The latter expects his purchase orders to contain three fields: part number, quantity and delivery date. The West Coast partner expects her purchase orders to contain part number, quantity, delivery date and preferred shipping carrier. Thus they each have different definitions of a purchase order. How will they converse? While this particular problem doesn't seem that complicated, multiply the number of partners by 10 or 100, each with his or her own definition of a purchase order. Now we have a problem!

The naive approach to dealing with this problem would be to have our Java code deal with the individual partners in a special way. The problem with this approach is that each partner that requires special information forces the modification of the Java code used to implement the business model.

JAVA AND THE HOW

The ideal solution is to create a generic Java program that doesn't have to deal with the individual requirements of each partner. This can be done using XML. A core exchange format can be set up between you and your partners, and the individual information required by each partner can be abstracted in a properties file. The file will be responsible for matching additional information to a specific partner. In this particular scenario each partner will deal with the information he or she understands; the remainder of the information will be ignored. As new partners join your "circle of friends," the only information that needs to be modified is the properties file and the XML data file. This is where the power and flexibility of the XML data format complements the power and flexibility of the Java runtime environment. Furthermore, a neat side effect is that the properties file could have been written using XML.

XML Documents and Dynamic Class Loading

When people talk about XML for data representation, the most basic concept they refer to is a document structure with data. This structure, similar to a populated C structure, outlines a tree whose nodes describe the content found on the leaves. Simple documents don't contain any behavior that defines how to access the content on the tree. Thus an XML document can be thought of as a data object with accessor methods. This idea can be heavily leveraged to implement exchange formats for transaction protocols. More complex XML documents leverage the concept of mobile agents to provide behavior to XML documents. This approach leverages URL links embedded inside the document as object repositories from which functionality can be downloaded over the Web and used to process specific document tags. It is here that XML leverages the power of Java to extend its data model to add behavior. The Java code contained in the URL links is downloaded via the URL class loader mechanism contained in the Java platform. Once the class bytecodes are downloaded over the Web, a class object is created and temporary object instances are created and used to evaluate the information contained inside the XML file. This enables the dynamic extension of program behavior. Another way in which Java components can be leveraged is to send mobile agents to evaluate information stored inside XML files by analyzing the tags contained inside the document.

How Does XML Speak Java?

By now I hope you're convinced about the complementary roles of Java and XML. Let's move forward to explain how the two technologies come together. XML–Java parsers hold the key to the answer. XML data parsers written in Java provide two standard interfaces blessed by the W3C:

- ***Document Object Model (DOM)***: The DOM provides a mechanism that allows users to access the information contained in the document in a tree fashion. Traversing the tree is left to the application writer. Users of the DOM normally care about the hierarchy and structure of the document.
- ***Simple API for XML (SAX)***: The SAX provides an event-driven method for traversing the information in the document. Application writers can register callbacks that are invoked when the beginning and ending of a tag are parsed. Once inside the callback, the program is able to discriminate against the tag information. Users of the SAX care about specific tags inside the document, not necessarily its hierarchy.

I'd like to illustrate the power of the SAX and DOM interfaces by providing an example. Imagine you and your partners agree on a format for exchanging purchase requests. The information needed by all partners in order to process the purchase request is buyer name, buyer address, order number, product ID, product name, quantity, delivery date and requested price. This particular hierarchy identifies the root element of the document as the PurchaseRequest. It contains three additional elements called BuyerName, BuyerAddress and OrderNumber. Multiple

(continued on page 12)

XML in Tango 2000

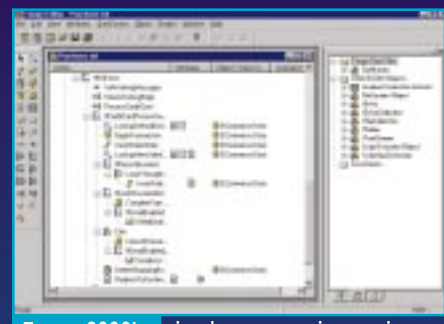
WRITTEN BY TOM OTVOS

Tango 2000 is a singularly powerful and easy-to-use tool for creating dynamic, intelligent Web sites that are integrated with popular database systems. Unlike other application servers that take a simplistic "mail-merge" or page-centric approach to page generation, Tango 2000 uses a visual programming metaphor that enables users to create powerful Web applications in record time without having to be programming gurus.

A particularly powerful feature of Tango 2000 is its support of XML. In basic terms, XML is supported as a first-class application data type, enabling the developer to create XML (or, more precisely, DOM) variables that represent complex hierarchical data structures and access, manipulate and stream them with ease. An important point is that any XML structure may be used; the developer isn't limited to a single DTD (such as WDDX).

A good question at this point would be: "Why should I want to create and manipulate XML structures in a Web application?" One reason is to enable a developer to separate business logic from presentation. If you consider a reasonably complex Web application like a shopping basket or portfolio tracker, the logic involved in accessing and processing data (the business logic) can be quite intricate. This typically involves grabbing data from one or more data sources and applying various business rules. The presentation logic on the other hand is focused solely on putting the data in front of the user in a visually compelling way. In large applications these tasks are often done by different people.

It's pretty much axiomatic that most programmers don't know much about user interface design. Therefore, it's a safe bet that the person developing the business logic isn't going to be a hot HTML whiz. Likewise, the presentation-capable person won't necessarily know SQL from Java. Enter XML, which provides a lingua franca between these two camps. As long as both agree on an XML structure as an intermediate representation of the data (i.e., an XML DTD), the business logic programmer can use arbitrarily complex logic to create the data without affecting how the visual designer decides to present that data to the user. The flexibility of XML to describe complex hierarchical data simply makes it an ideal choice. Even if the business logic and presentation are created by the same person, taking the critical step of separating logic from presentation makes for a much more maintainable application. This is particularly difficult to do in page-centric Web application environments.



Tango 2000's visual programming environment makes building XML applications easy.

A second – and more compelling – reason to expose XML functionality in Tango 2000 is to empower developers to capitalize on the emerging use of XML as a form of "Web EDI." Because of its plain-text, self-describing nature, XML is rapidly becoming the de facto standard for transferring data between remote and heterogeneous systems. From Microsoft's BizTalk initiative to ERP vendors such as SAP's Business-to-Business Procurement solution, XML is playing a central role in linking systems together. Tango 2000's focus on open and unlimited XML structures provides an opportunity for developers to create "integration solutions" – solutions that combine heterogeneous business systems with ease and flexibility.

Consider Tango's role in a hypothetical portfolio tracking application. A user could view his or her portfolio via a Web browser connected to a central Tango application. To make a trade, the user would simply place the order through the Tango application, which would then forward the trade details to a secure server at the brokerage via XML and HTTP. Upon confirmation of the trade (which the Tango application could automatically poll using another XML request), the client's portfolio would be updated. Skeptical? Well, if you do online banking, this is happening already. The next time you connect to your bank using Quicken and download your transactions, save them to a file and have a look. That's XML at work.

Through an intuitive interface, Tango 2000 simplifies the reading, manipulation and streaming of XML data. Moreover, unlike some competing application servers that impose a fixed DTD on your XML, Tango allows the developer to choose precisely what data is sent over the wire. This means that existing back-end servers that already speak XML don't need to be rewritten to allow a Tango application to access them. That flexibility, coupled with Tango 2000's programming model, makes it a good choice for creating XML-centric Web applications easily.

AUTHOR BIO
Tom Otvos is director of research for Pervasive Software.
tom.otvos@pervasive.com

OrderNumber tags can be added to the document to represent various orders from the same buyer. BuyerAddress is made up of four elements: StreetName, City, State and Zip Code. OrderNumber is made up of four elements and one attribute. The four elements are called ProductID, ProductName, DeliveryDate and RequestedPrice, and the attribute name is Quantity. Although the Quantity attribute could have been expressed as an element, for our particular example it's more advantageous to define it as an attribute because it can be directly manipulated by the SAX API inside the callbacks attribute list of the OrderNumber tag. Figure 1 illustrates the document hierarchy. The XML document format is shown in Listing 1.

Some partners may take the orders, process them and notify senders of the status of their order. These partners parse the information contained in the document in a batch manner and create objects that are used by their purchase order systems. Based on the information contained in the document, the purchase order system might create three objects: a purchase request object, a buyer object and an order object. The purchase request object contains the buyer object and a list of order objects. The DOM interface is the correct mechanism to facilitate the creation of these objects from the XML document. Listing 2 shows the use of the DOM Java API to retrieve the document information needed to create the buyer object.

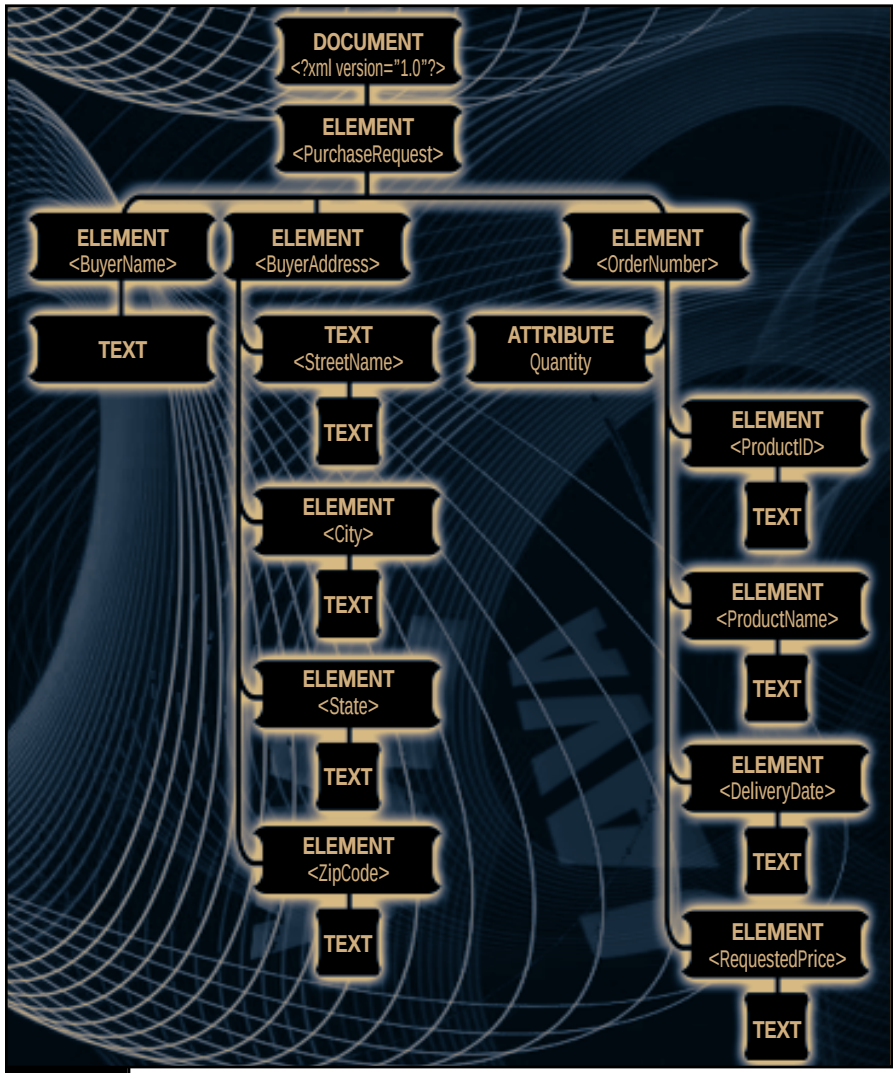


FIGURE 1: Purchase Request XML Hierarchy Definition

Some partners may want to evaluate requests whose item quantity is greater than or equal to 500. To facilitate processing, the application programmer may wish to evaluate the "Quantity attribute in the 'OrderNumber'" tag independent of any other information in the document tree. In this case we use the SAX interface, which, among other things, allows us to register a document handler as a callback object that's triggered when a document tag is found. When the tag being processed is equal to "OrderNumber," the quantity attribute will be evaluated against the quantity rule. In this scenario it's irrelevant that the "OrderNumber" tag is contained inside the "PurchaseRequest" tag. Listing 3 shows the use of the SAX Java API to capture the "OrderNumber" tag from the document information and evaluate its Quantity attribute.

In this particular example orders fewer than 500 items will not be processed by the system. Those orders greater than or equal to 500 will be processed using the DOM API. However, in this case the SAX API will allow the application writer to filter the information and not overload the system with unprofitable requests.

The definition of a standard representation of the purchase order document enables the various partners to manipulate the information as they see fit, independent of each other. This flexibility can be extended by allowing sophisticated partners to add additional tags into the document hierarchy. As long as the main tag dependencies are kept, the sophisticated partners will be capable of leveraging the additional information on their transactions. Additional tag examples can be a "DeliveryDateOffset" tag that allows a partner to identify a range of days from the "DeliveryDate" tag by which the order can be supplied. If the information is present in the purchase request, the partner can leverage it. If it's not present, it can be ignored by the partner system.

The URL class-loading capabilities of the Java 2 platform supplement the XML data model by allowing a document to contain behavior in addition to data. This is accomplished by embedding URL links to Java classes inside a document. Class loading coupled with reflection form a powerful mechanism that allows Java programs to dynamically download functionality from a partner Web site in order to process new XML tags. However, this particular mechanism requires an adapter-based framework similar to the Beans model that allows application developers to dynamically define interactions between their legacy systems and the newly downloaded functionality. (This mechanism will be covered in a separate article.)

Conclusion

In this article we discussed the advantages of marrying the XML and Java technologies. XML is to Java as cream is to coffee; it makes the coffee drinkable. While Java by itself provides a great deal of dynamic behavior through its dynamic class loading and reflection mechanisms, by itself it's not the best way to deal with data format issues. XML takes Java to the next level by providing a flexible and extensible tag definition environment that is machine independent. Java applications coupled with XML data formatting are more capable of adapting to data format changes in a generic, nonprogrammatic way. This increases time to market and gives developers the ability to react more quickly to market changes.

Author Bio

Israel Hilerio, a member of the technical staff at i2 Technologies, Dallas, is a Sun Certified Java programmer with 10 years of programming experience, including three and a half in Java. He has Ph.D. and MS degrees in computer science and a BS in computer engineering.

israel_hilerio@i2.com

Listing 1

```
<?xml version="1.0"?>
<PurchaseRequest>
  <BuyerName>TonyElectronics</BuyerName>
  <BuyerAddress>
    <StreetName>909 E. Las Marias</StreetName>
    <City>Dallas</City>
    <State>Texas</State>
    <ZipCode>75090</ZipCode>
  </BuyerAddress>
  <OrderNumber Quantity="500">
    <ProductID>PC102</ProductID>
    <ProductName>Siega 400MHz Pentium PC</ProductName>
    <DeliveryDate>November 19, 1999</DeliveryDate>
    <RequestedPrice>$1200</RequestedPrice>
  </OrderNumber>
</PurchaseRequest>
```

Listing 2

```
Document doc;
TXElement root;
Buyer B;
Parser p = new Parser("XMLParser");
try {
     FileInputStream file = new FileInputStream(documentName);
     doc = p.readStream(file);
     root = (TXElement) doc.getDocumentElement();
     TXElement name = (TXElement) root.getElementNamed("Buyer
     Name");
     TXElement address = (TXElement)root.getElementNamed("Buy
     erAddress");
     TXElement street =   (TXElement)address.getElement
     Named("StreetName");
     TXElement city = (TXElement)address.getElement-
     Named("City");
     TXElement state = (TXElement)address.getElement-
     Named("State");
     TXElement zip = (TXElement) address.getElementNamed("Zip
     Code");
     B = new Buyer(name, street, city, state, zip);
}
catch (java.io.IOException e) {
         e.printStackTrace();
}
```

Listing 3

```
try{
      Parser parser = ParserFactory.makeParser(parserClass);
      saxTest handler = new saxTest();
      parser.setDocumentHandler(handler);
      parser.setErrorHandler(handler);
      parser.parse(args[0]);
      System.exit(0);
}
catch (Exception e) {
      e.printStackTrace();
}

public void startElement(String name, AttributeList atts) {
      if (name.equals("OrderNumber"))
      {
        int qty = Integer.parseInt(atts.getValue("Quantity"));
        if (qty >= 500)
        {
             processOrder(filename);
        }
   }
}
```

▼▼▼▼▼▼ CODE LISTING ▼▼▼▼▼▼

The code listing for all articles can also be located at www.JavaDevelopersJournal.com

Excelled Developers

A sad story of failure and frustration

WRITTEN BY
ALAN WILLIAMSON

very time I come around to writing this, I have this fear: What the hell am I going to write about? Then I sit back and have a think of what has happened in the last month and what is likely to happen in the forthcoming month and something usually presents itself. Fortunately, Mother Nature has something quite exciting up her sleeve for me to touch on this month, so I thank her for that. This is the month of the eclipse of the sun – no, nothing to do with McNealy and associates. This is *the* SUN.

Last month I was all excited with the afterglow of JavaOne. If I droned on a bit too much about that, don't worry. This month will be JavaOne free. Doh! I've just gone and mentioned it there. Sorry.…Sorry.

So what has been happening this month? Well, we've been having lots of fun with Java and Microsoft, and have also seen the launch of the UK's first Java Servlet hosting ISP. But before all that, let me tell you about my new wee toy.

Yes, I am very excited. For years I've been fighting the urge to get a portable computer. As you know, I do quite a bit of traveling. This very article has been penned on a variety of people's computers. I even had to resort to the old pen and paper on one dark occasion. I'm a bit like the magpie in that respect. Not having a nest of my own, I tend to flock to others for such "nest" requirements. The resistance has been for any of a number of reasons. One, I was never a fan of the size of them. When you're walking about, it's always obvious that you're a "carrier," and this never really appealed. Considering the career path I've chosen, I'm not a big fan of the technology gadgets in which we insist on entrusting more and more of the responsibilities of our daily lives. Besides, if I had a portable, others might be under the illusion that I actually do coding on the move. I don't think so! I don't even wear a wristwatch. Never found the need for one.

Of course, I have gone through all the fads of personal organizers – my desk drawer is a testament to that. Therein lies a graveyard of old Psions and Apple Newtons, and there's even an old Casio address book knocking about. But I get bored with them. Sure, it's a great novelty to carry these things around, and let's be honest, there's a certain street-cred to be had when displaying your wares in the high street or office. When I had my Newton, it was a great way to talk to people as they would come up in bars and restaurants and go ooh and ah.

But more recently, as my company grows and expands, I find myself doing more and more documentation, and keeping on top of e-mail has become paramount. So the need for a portable PC raised its ugly head again. I had a look around at the current offerings, and not a single unit caught my eye. I was still bored! But then I remembered a machine I spotted in Tokyo last year and thought, How cool is that? It was the Sony Vaio PictureBook, the one that's half the size of normal portables and has a camera built into the top of the screen. It only recently became available on the market, and when it did, I was bowled over by the price of it: so cheap. So I ordered one, and a couple of days later it was delivered. I love it. It's a full PC with Windows 98 and has my entire development environment on it with all my documentation tools. But it's small, with a keyboard just the right size. Any smaller, it would be awkward to type documents of any real size. It's no bigger than the Apple Newton, so I can carry it about with my other crap without the need for a separate "Targus" bag.

Excelled…

My developer's background is with the early Windows API. For reasons best known to me at the time, I wanted to be a Windows programmer. I wanted to create great programs that would blow people away. But I was constantly frustrated about why official Microsoft products always ran better than mine and seemed to have access to features I couldn't find any documentation for. Then I discovered the ancient art of the undiscovered APIs. An experienced developer who mentored me took me through some of these real beauties that aided and abetted my cause. Well, this was a practice I assumed was being downplayed as I moved through the Java community. After all, we've heard many of the big software vendors complain about Microsoft's undocumented APIs, and how this unfairly creates an uneven playing field. But being solely in the Java universe, this is an argument I've been able to follow as closely as anyone would want.

But my company has been involved with a project that called for some of this knowledge. One of our requirements was the ability to read an Excel spreadsheet file and be able to manipulate it. I had performed something similar many years ago under Windows and C, and recall having lots of wonderful fun when trying to write an Excel version 4.0 file.

So it was with a certain amount of trepidation that we ventured into the world of writing a class that could read Excel 97 files. As the first port of call, we researched the Microsoft site, looking for information on the structure of these enigmas. At first call it seemed rather easy. All the information seemed to point toward a rather simple HEAD/DATA record format. You simply read the header information, which then paves the way for the layout of the data section.

Simple. Creating a Java class that could perform this didn't seem too problematic. After creating a core structure that would iterate through all the records in a file, we felt confident we would have this one done 'n' dusted within a couple of days. Well, you know the feeling you get when you're coding, that what you're working on is a work of art. It will run first time and the optimism you feel allows you the luxury of a false sense of security.

We compiled and then ran. Failed! Not a single record was read. Ho hum, guess we must have missed something silly. Looked through the code; nothing major was highlighted. Must be something we missed, but nothing obvious was coming forward. Feeling a little discouraged, we resorted to opening up the original Excel file with a hex editor to see if we could read it manually.

Nothing looked familiar. But after, say, the first 512 bytes, the format looked characteristic of what we were looking for. So we modified our class to skip forward the necessary bytes and lo and behold! The code started to churn out Excel records. But then it failed as it approached the end of the file. How frustrating.

Nothing in the Microsoft documentation pointed to or even hinted at this strange behavior. We looked through the postings in the newsgroup via what has to be the best research tool in the world: www.deja.com. We found many others that shared our problem, and even one guy who said, "Oh, you don't want to be doing that." Very helpful, thank you.

The problem seems to stem from when we encounter a worksheet boundary. From there on in the file becomes unreadable. We're still unable to read multiple worksheets, and at the present moment we have a class that reads a single worksheet. If anyone has successfully unlocked the secrets of the Empire's Excel spreadsheet, please don't hesitate to e-mail me. Would love to hear from you.

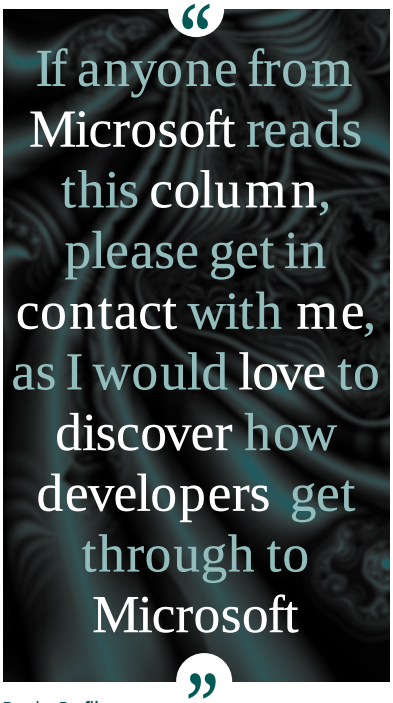
We of course tried to e-mail Microsoft with our query, and what a well-invested effort that turned out to be! Having e-mailed a number of Microsoft channels, it's now the third week and we've still heard nothing. If anyone from Microsoft reads this column, please get in contact with me, as I would love to discover how developers are to get through to Microsoft. The likes of Sun and Oracle have opened their doors to developers, and I am wondering whether it's a case of not knowing where this mythical door is located, or does the door, like the mythical Scottish town Brigadoon, appear only every 100 years. Answers in an e-mail, please.

AUTHOR BIO

Alan Williamson is CEO of n-ary (consulting) Ltd, the first pure Java company in the United Kingdom. A Java consultancy company (www.n-ary.com) with offices in Scotland, England and Australia, they specialize solely in Java at the server side. Alan is the author of two Java Servlet books and contributed to the Servlet API.

Mailing List

I'm sure we're not the only ones to have hit this sort of closed-door syndrome. How many of you have had to wrestle with other Microsoft formats? I'd love to hear from you. Come and chat on our mailing list and share your views with the rest of us. To join, send an e-mail to listserv@listserv.n-ary.com with subscribe straight_talking-l in the body of the e-mail. From there you'll get instructions on how to participate on the list.



Reader Profile

A wee while ago I briefly touched on the makeup of you all and recounted the weird and wonderful places in which you've been reading this fine piece of literature. Well, many of you have e-mailed me with some strange places, and I think the current leader in this rather unique field has to go to Giuseppe Persiani, an Italian Java developer. Giuseppe e-mailed me about a number of issues I raised and then tagged on the bottom where he was when he first read this column. It was just before he saw his daughter for the first time on an ultrasound in a clinic. Whoa. I assume, of course, that he took ***JDJ*** in with him. I can't imagine ***JDJ*** forming that pile of communal magazines we all nervously thumb through while we patiently wait our turn. If I'm wrong and it was, what a cool clinic! – it gets my vote.

Speaking of that, we live in a farming community and the types of magazines we have are *Farmers Weekly* and *Scottish Farmer*. What sorts of magazines are available in the deepest Silicon Valley clinics? Would love to know if any techy journals are among this pile. Let me know, please.

Salute of the Month

In this article I'd like to take my hat off to someone who over the last month has gone above and beyond the call of duty. This month I am going to stay at home and salute two of my developers. I had disappeared way up to Glasgow for the weekend, and came home late Sunday afternoon to discover the office open. I thought, hmmm, how strange. I wonder who's in today? I then proceeded to enter the premises and discovered Keith and Murray working away, developing Java. It was a hot Sunday afternoon, and we had a project that looked as though it was going to overrun. These guys had the forethought to try to get a lot of the work done earlier as opposed to leaving it to later. To celebrate this act of foresight, we then proceeded down to our local for an evening's drink or two, and it turned out to be a very good night. So Keith and Murray, I thank you.

Book Review

This month I finished the story behind Netscape from Jim Clark. It was actually a very short book when all is said and done, and most of it was filled with the observations of Jim Clark comparing Netscape to the way his old startup, SGI, ran. It was a very interesting read, and it really opened my eyes to the history of Netscape. My only criticism is that it didn't go into enough detail. This was another book about a company that suffered from having no sales, then you turn a page and they're suddenly having millions of dollars worth of sales. Most frustrating. Again no explanation of how they grew that side of the business, and no information on how bagging that first revenue check really felt. Are these people becoming complacent and forgetting the simple joys of starting a business? I know that from my point of view I'm more interested in the early days than how they are battling the likes of Microsoft. Would love to know how these chaps operate around the birth of a company. That said, excellent book, and highly recommended.

• • •

Spending time with some of you Americans is beginning to scare me – and more important, affect me. I've found myself saying "I guess" an awful lot more. I can't seem to catch it in time, and once it's out in the open you have to run with it. But inside I'm kicking myself, and mentally noting not to say it again. So before I'm tempted to utter it again, I shall bid you farewell, and look forward to speaking to you next month.

Have a nice day y'all! Noooooooo!

alan@sys-con.com

THERE'S A NEW PLAYER IN TOWN
IS EVERYONE JUMPING ON THE BANDWAGON?
XML in Oracle8*i*

WRITTEN BY MARTIN BOYD

A year ago developers were just learning that XML stands for eXtensible Markup Language. Six months ago CIOs started taking an interest in XML and smart developers started buying "Introduction to XML" books. Now "XML for Managers" books are becoming popular and it seems like everyone is trying to figure out what they can do with XML and what infrastructure they'll need to support it.

Why XML?

There are many reasons to use XML as an enabling data format and many potential applications have been suggested, from improved Internet searching to defining configuration files. However, the two uses of XML that appeal most to both CIOs and developers are the exchange of business data between applications and customized presentation of data.

- ***Exchanging business data between applications***: Application developers need to tie enterprise systems from different vendors together within an organization to enable, for example, order-processing applications to pass information to inventory management and for both systems to access the customer database. Using XML, a loose integration is possible at a fraction of the effort of traditional EDI. With XML, businesses don't need to replace or rebuild their applications; rather they will simply begin to XML-enable the data and systems they already have.
- ***Customized presentation of data***: Different browsers and devices, such as PDAs (personal digital assistants), cell phones or pagers, have unique display formats. Using an XSL processor, on either the client or server, XML data can be transformed into the appropriate format for display on any client, completely insulating the business application from the constraints of the output device. This separation of form from content is key to building any flexible and extensible information system.

How Do Databases Fit In?

For a short while in the rapid evolution of XML there was a notion that specialized XML servers and object databases would be the natural storage point for XML documents and data. But then a question arose: "So where does all this XML data come from?" The answer, of course, is: "From the relational databases that underpin the majority of business applications in use today." For most XML applications, the creation and ongoing maintenance of a separate "XML repository" is an unnecessary development expense. Instead, all most companies need to do is take information from existing databases and render it as XML so it may be shared and consumed more easily. Of course, if the relational database in question has object capabilities, such as Oracle8*i*, then the task of rendering and updating data as XML is greatly simplified.

If the key requirement is to be able to efficiently read and write XML to and from the database, developers should look hard at Oracle8*i*. Designed from the ground up to be the database for the Internet, it includes native support for Internet standards such as Java and XML. In fact, at the heart of the Oracle8*i* database is a highly scalable Java engine tuned for server-side Java application development, giving Java the ability to scale to thousands of heavy-duty concurrent connections. In addition, Oracle's XML capabilities are so much a part of the database that Oracle8*i* can run its XML components for Java completely within the database to deliver the scalability required by today's Internet applications.

Let's look at the tools and techniques used to read and write XML in Oracle8*i*.

Reading XML from the Database

The ability to model or materialize a database object, such as an insurance claim form, in XML requires infrastructure components that can use SQL to request the object and transform it into an XML-formatted document or datagram. In the example in Figure 1, a claim form is defined as an object in the database using the relational tables shown.

This object can be materialized in XML as follows:

```
<?xml version="1.0"?>
<CLAIM>
 <CLAIMID>123456</CLAIMID>
 <FILED>1999-01-01 12:00:00.0</FILED>
 <CLAIMPOLICY>
  <POLICYID>8895</POLICYID>
  <PRIMARYINSURED>
   <CUSTOMERID>1044</CUSTOMERID>
   <FIRSTNAME>John</FIRSTNAME>
   <LASTNAME>Doe</LASTNAME>
   <HOMEADDRESS>
    <STREET>123 Cherry Lane</STREET>
    <CITY>San Francisco</CITY>
    <STATE>CA</STATE>
    <ZIP>94100</ZIP>
   </HOMEADDRESS>
  </PRIMARYINSURED>
 </CLAIMPOLICY>
  <DAMAGEREPORT>
  The driver lost control of the vehicle.
  This was due to <CAUSE>faulty brakes</CAUSE>.
 </DAMAGEREPORT>
  <SETTLEMENTS>
  <PAYMENT id="0">
   <PAYDATE>1999-03-01 09:00:00.0</PAYDATE>
   <AMOUNT>7600</AMOUNT>
   <APPROVER>JCOX</APPROVER>
  </PAYMENT>
 </SETTLEMENTS>
 </CLAIM>
```

This XML document preserves the data relationships set up by the database schema for the insurance claim object. The following illustration was built using the Oracle XML Utilities and shows how a simple Java servlet (the XSQL Servlet, available for download on the Oracle Technology Network) can create this output.

The XSQL Java servlet can be loaded into a database that supports Java, such as Oracle8*i* shown in Figure 2 or a middle-tier server such as Oracle Application Server or other Web servers that support servlets. Let's take a look at the XSQL Servlet structure in detail.

Oracle XSQL Servlet for Java

The XSQL Servlet is a Java program that leverages the capabilities of Oracle8*i* to assist in producing dynamic XML documents from one or more SQL queries of data objects. It does this by processing a .xsql file, which is simply an XML file with a specific structure. The following is an example of a .xsql file that searches for insurance claims submitted by Mr. Doe. Note that the query may be written as regular SQL or, as in this case, using object dot notation.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="claim.xsl"?>
<query connection="xmldemo">
  select value(c) as Claim from insurance_claim_view c
  where c.claimpolicy.primaryinsured.lastname = 'Doe'
 </query>
```

The servlet uses Oracle's XML Parser to process this file and pass any XSL processing statements to its internal XSLT Processor (which performs transformations according to an XML stylesheet) while passing the parameters and SQL statements between the <query> tags to the XML SQL Utilities. Results from those queries are then received as either XML-formatted text or a JDBC ResultSet object. If necessary, the query results may be further transformed into any desired format using the built-in XSLT processor.

The XML Parser is used by the XSQL Servlet to parse the .xsql files as well as to receive and render as necessary the output from the XML SQL Utilities. As with the servlet, the Parser can be loaded into the Java VM (virtual machine) in Oracle8*i*, run from a middle-tier server such as Oracle Application Server, or run on the client as a JavaBean. Through its DOM and SAX support, it can accept the ResultSet from the XML SQL classes and provide an XML object tree or text stream to the servlet for output.

XML Transformations

The XML Parser's XSLT support isn't limited to transforming one XML document into another. These XML documents can also be used as data messages between heterogeneous databases, with XSLT providing the translation from one message format to the other. While this capability is useful for electronic data interchange, it can also be used to render documents in HTML and other text-based formats. Creating a stylesheet that is simply an HTML page with the appropriate XSL processor statements at the locations where the XML data is to be displayed can dynamically create very effective HTML pages.

For example, using the insurance claim XML document, we can create an HTML page to render selected data elements, initially using dummy data as page-formatting placeholders. Once the HTML is satisfactorily formatted, the static data may be replaced with XSL processor statements, as below:

Data	XSL Statement
JCOX	<xsl:value-of select="claim/settlements/payment/approver"/>
7600	<xsl:value-of select="claim/settlements/payment/@amount"/>
123456	<xsl:value-of select="claim/@claimid"/>

Writing XML into the Database

Its self-describing object capability makes XML a compelling technology for e-business and other data interchange applications. Using application generation utilities such as Oracle's XML Class Generator for Java, Web-based applications can create XML documents or datagrams to be stored in databases.

There are two ways to store XML-tagged data – as a single object in a CLOB or BLOB with its tags intact or distributed across a set of tables. Distributing across tables maps XML data into the relational schema. Oracle8*i* can accept an XML document as a single object and, through its *inter*Media text option, can actually search for and return data based on its XML tags. Using our insurance example, the following SQL statement searches for all settlements approved by JCOX and uses the CONTAINS SQL function to find those where "faulty brakes" were the cause.

```
SELECT SUM(Amount)
  FROM Claim_Header ch, Claim_Settlements cs,
       Claim_Settlement_Payments csp
WHERE csp.Approver = 'JCOX'
AND CONTAINS(DamageReport,'faulty brakes WITHIN cause')>0;
```

While this is quite useful for unstructured data such as the accident description on our claim form, structured data is better served by storing it in tables without tags where it can be easily updated and queried. Oracle's XML Parser with its XSLT transformation component can parse an XML document submitted to the database and, based on an XSL stylesheet, create the appropriate SQL INSERT and UPDATE statements to store the XML-tagged data in the database.

Referring back to our insurance claim document, the following stylesheet fragment could create the Payment table entry after XSLT processing:

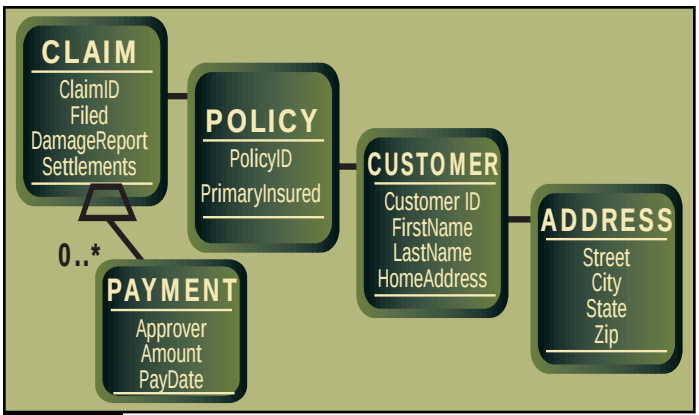


FIGURE 1: Claim form defined as object in database using relational tables above

```
insert into PAYMENT values
  ('<xsl:value-of select="claim/settlements/payment/approver"/>',
   <xsl:value-of select=""claim/settlements/payment/@amount">,
   <xsl:value-of select="claim/settlements/payment/@paydate"/>);
```

Alternatively, the OracleXMLSave class within the XML SQL Utilities is able to put back the XML document into any database table or view. It maps the tag names to the column names and handles structured types, collections and references appropriately.

Essential XML Building Blocks

Applications such as the XSQL Servlet described above make use of lower-level XML components including the Oracle XML Parser, Class Generator and Utilities. The Java versions of these components can be loaded into the Oracle8*i* server, run from a Java VM on a middle tier such as Oracle Application Server or on the client. They are available along with working code samples through the Oracle Technology Network to all registered members. Registration is free (http://technet.oracle.com/tech/xml).

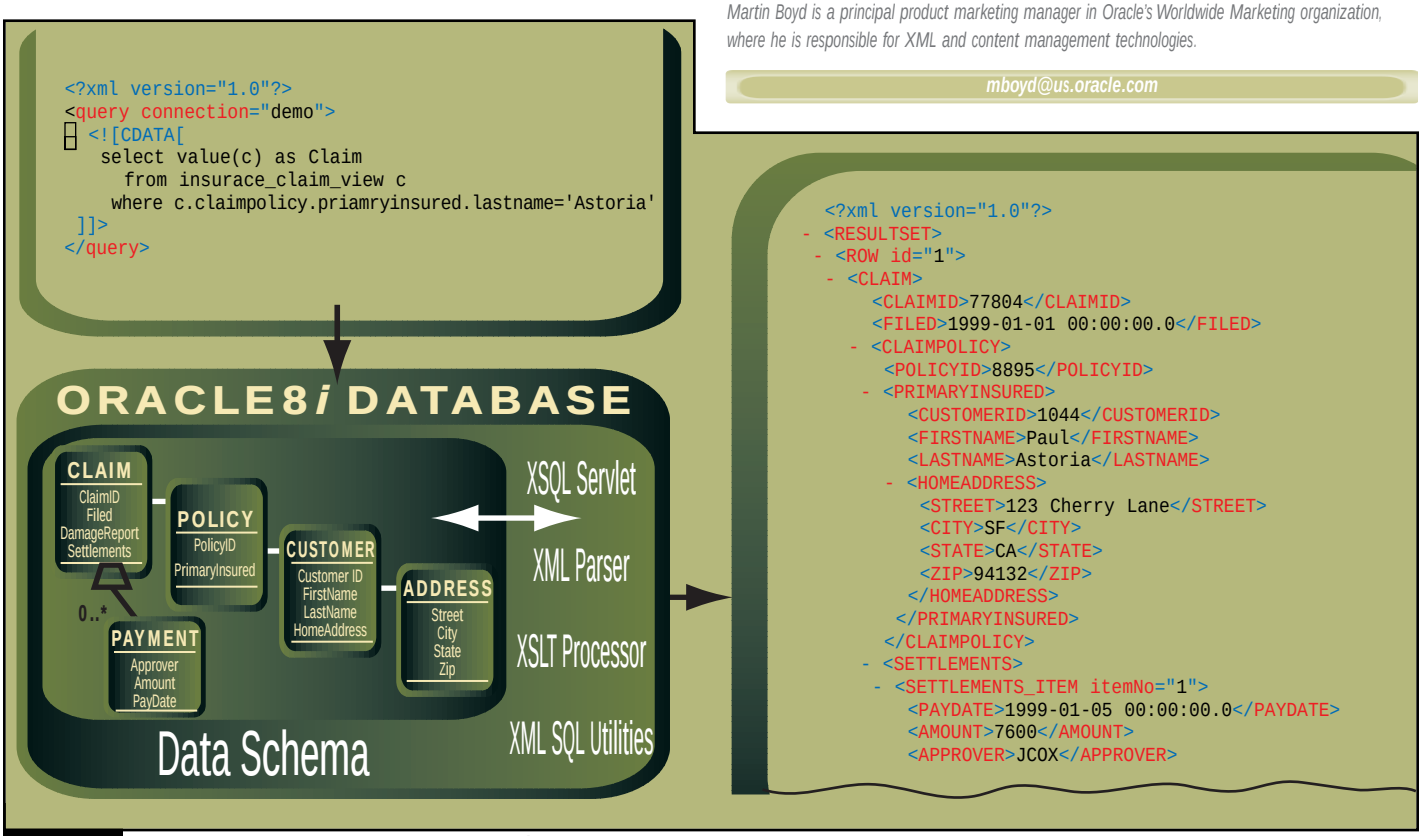


FIGURE 2: The XSQL Java servlet loaded into Oracle8*i* database

- ***Oracle XML Parser***: Available in Java, C, C++ and PL/SQL. These parsers are available on Windows, Linux, all flavors of UNIX, and all the other platforms on which Oracle runs. The XML Parser for Java is fully compliant with the DOM Level 1 specification supporting validating and nonvalidating modes with an integrated SAX interface and support for Namespaces. Performance has been optimized through various strategies, such as caching DTDs and incorporating internally the XSLT processor for transforming XML documents based on CSS and XSL stylesheets.
- ***Oracle XML SQL Utilities for Java***: These utilities take a SQL or SQLJ query and return the results in XML as text or DOM trees. SQL queries from the XQSL Servlet, Java stored procedures or even the command line can be sent to the database and will generate the results formatted as XML based on the internal structure of the schema.
- ***Oracle XML Class Generator for Java***: The XML Class Generator for Java is a utility that creates Java classes from DTDs to enable the programmatic construction of XML documents. The Class Generator works in conjunction with the Oracle XML Parser, and boasts complete character set support and optional validation mode for ease of debugging. Documents created by these classes are fully compliant with the XML 1.0 W3C standard.

Standards Support

There's a lot of confusion and hype in the market. With all the current interest in XML, many vendors claim to be developing the "next XML standard" for any number of different applications or business types. Whether focused on the XML language itself or its DTD data standards, these efforts do little to promote clarity and only spread F.U.D. – fear, uncertainty and doubt – among decision makers. Regardless of the shakeout from the definition game, Oracle is committed to supporting open, vendor-neutral standards for XML. As an active representative on many W3C working groups and a founding member of XML.org, a vendor-neutral organization working to define the data standards, Oracle is working hand in hand with vendors and customers to ensure that, whatever that standard, customers will have the best technology platform to seamlessly integrate and run their applications.

Author Bio

Martin Boyd is a principal product marketing manager in Oracle's Worldwide Marketing organization, where he is responsible for XML and content management technologies.

mboyd@us.oracle.com

The Mark(et) of XML

Is it the next killer app?

WRITTEN BY
AJIT SAGAR

A few months ago, at JavaOne, I discussed the possibility of starting an XML publication with the folks at SYS-CON Publications. Two questions came up: "Is it as big as Java?" and "Are there any real products out there?" Both are valid.

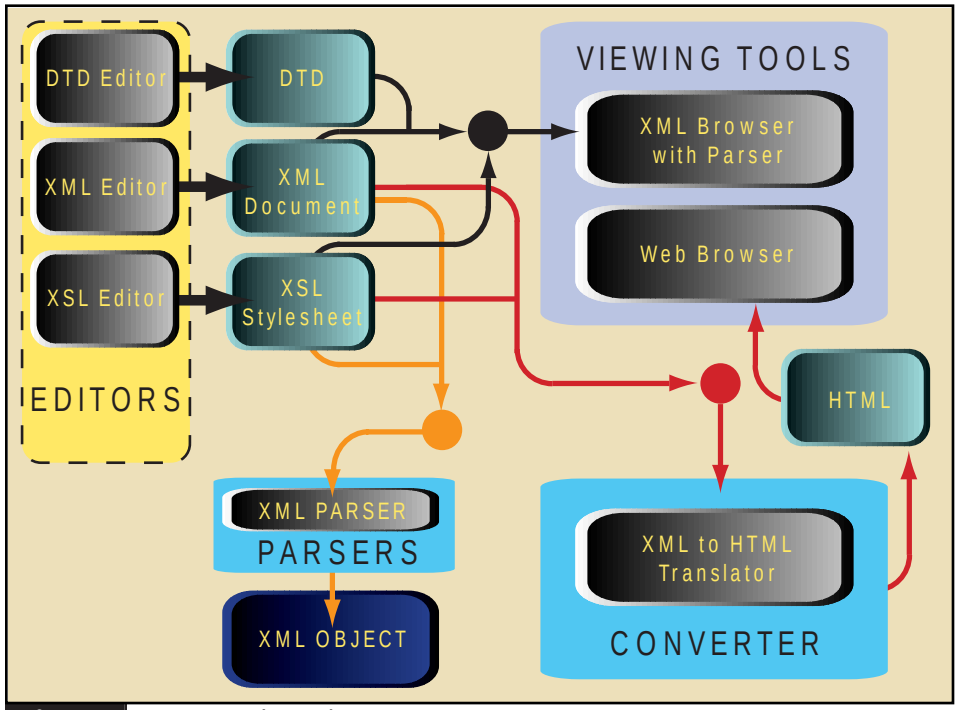


FIGURE 1: XML presentation tools

The first question is the more difficult to answer. XML is a standard. Java is a platform. Is XML as big as Java? Or C++? Or EDI? Or HTML? For one thing, XML isn't a programming language. It isn't a software platform. It isn't a Web presentation language. In some sense XML may be seen as an Internet-enabled version of EDI. However, at its most basic definition, it is a markup language. One of the primary purposes of a markup language is to represent data via a tag-based scheme. XML stands for eXtensible Markup Language. This means that not only does XML allow data representation via tags, it enables the definition of the tags themselves.

Thus XML may be seen as a means of data definition and representation. Since data is an integral part of all computing environments (indeed, it is their raison d'être), XML transcends programming languages and computing platforms. In that sense it is "bigger" than Java. XML holds the promise of being the cross-language, cross-platform common data format. Its ultimate goal would be to become as ubiquitous as HTML in vertical industry segments.

The second question addresses the promise of the "common data format" as it stands today. The market for XML is evolving rapidly. Some products have already matured to industrial strength. Others are in the process of maturing. And many are still in their inception stages. However, the XML products in the market today run a wide gamut and span several tiers of the computing industry. This leads to confusion about how to use XML and in what areas of computing. Currently there is a wave of euphoria about XML in the industry. People are referring to it as the "silver bullet" for e-commerce. While this may be an overoptimistic definition, it's true that almost anyone who's doing anything in the world of e-commerce is taking a serious look at XML.

This month in **e-Java** we'll take a look at XML product categories, vendors that are pushing XML as the enabler for e-business transactions, and XML organizations. As it's beyond the scope of this article to provide an exhaustive list of products and vendors, what I'll set forth here are randomly selected examples of vendors in the various product categories. I offer advance apologies to vendors not represented here, and encourage them to contact **SYS-CON** for representation in future publications.

Market Indicators

XML is yet another technology that's spreading like wildfire through the e-business market. Granted, a lot of it is hype. So what are the indicators for determining the maturity of the market? I'd like to present some of the unofficial indicators that appear in the industry when a technology starts staking its claim in the computing world:

- ***Committees:*** Technology committees both facilitate and hamper the evolution of a technology. Fortunately, the pros of the former usually outweigh the cons of the latter. Committees and consortiums promote standardization and reduce vendor bias. XML is a standard maintained by the W3C (World Wide Web Consortium). In addition, organizations like XML.org and OASIS (Organization for the Advancement of Structured Information Standards) add weight behind XML.
- ***Training:*** Software training organizations are a reflection of the need for particular software technologies in the market. Most consulting and training organizations involved in e-business training offer XML training.
- ***Online tutorials:*** Free online tutorials are an indicator similar to training. Several online tutorials on the use of XML are available on the Web.

PRODUCT CATEGORY	DESCRIPTION
Authoring tools	Editors for writing XML documents
Viewing tools	Browsers for viewing XML documents
Conversion tools	General and specific tools for converting to and/or from XML
Parsers/processors	Process XML documents to make them available to applications as an interface that the application understands
Tools/utilities	Various utilities that add features that sit on top of XML parsers/processors for different computing environments
Database systems	Provide persistent stores for XML documents; includes RDBMS and OODBMS products
Business-to-business servers	Enable application-to-application information exchange using XML
Content management systems	XML-based systems for document and content management

TABLE 1: XML product categories

- ***White papers:*** The good thing about white papers is that, although they're definitely biased, they serve to describe the general direction a particular technology is taking. When several major industry players start publishing white papers about a technology, that particular technology is one to be reckoned with. Almost any company that has a foot in the e-commerce door has a white paper that defines their direction regarding XML.
- ***Books:*** Take a look at the number of Java books that have flooded the market. In only four years well over a thousand books have been published with more to come. The XML book market is also growing rapidly. This again reflects a market need.
- ***Conferences:*** Conferences serve to bring IT managers, developers and vendors together. In addition to representation at several software and business conferences (including JavaOne), this year XMLOne, a SIGS conference focused solely on XML, took place in Austin, Texas, in May. Conferences are also scheduled for London and Santa Clara later in the year.

xml.orgs, xml.coms and xml zones

One indication of how serious vendors are in backing up a particular technology is the number of sites that subsequently start popping up. In the case of XML, XML.org (www.xml.org) is an industry Web portal operated by OASIS. XML.org is an industry consortium funded by a group of companies committed to establishing an open, distributed system for enabling the use of XML in electronic commerce and other industrial applications.

Several of the larger vendors have also started dedicating separate Web sites that focus solely on their affiliation to XML and the respective products. Here are some examples from the biggest software companies in the world:

- www.BizTalk.org: site of Microsoft's XML-based commerce server
- www.ibm.com/developer/xml: IBM's XML Zone
- www.oracle.com/xml: Oracle's XML Web site

XML Product Categories

The proof is always in the pudding. And actual products are the pudding in this case. Mature and evolving products are the best indicators of how far a technology has come. The XML products in the market today may be classified as:

- *Authoring tools (editors)*
- *Viewing tools (browsers, XSL tools)*
- *Conversion tools*
- *Parsers/processors*
- *Tools/utilities*
- *Database systems*
- *Business-to-business servers*
- *Content management systems*

Table 1 provides a brief description of each product category. Figure 1 illustrates the product categories that fit into the data presentation tiers. Figure 2 illustrates the product categories that fit into the business tiers. The remainder of this article discusses these categories and some representative products.

Authoring Tools

XML authoring tools include XML editors, DTD editors and XSL editors. XML editors are standalone products for creating and editing XML documents. Many XML editors are sensitive to DTDs and thus they can enable production of valid XML documents. Xeena from alphaWorks (IBM) and Adept Editor from ArborText are examples of XML editors.

XSL tools are used for creating, editing and processing XSL stylesheets. Examples of XSL editors are CUEXsl from CUESoft and XT from James Clark.

Viewing Tools

The XML viewing tools are the browsers that enable presentation of XML documents. XML browsers, generally driven by stylesheets, provide capabilities for viewing valid and well-formed documents. Browser support for XML varies among the different browsers. It's expected that Version 5.0 of Netscape Communicator as well as Internet Explorer will have richer support for displaying XML. Examples of XML browsers are Netscape's Mozilla, Internet Explorer 5.0, HyBrick from Fujitsu and XML Viewer from alphaWorks, IBM.

Conversion Tools

Applications using XML need to convert data from various formats to XML and back. This has given rise to a variety of general and specific tools for this conversion. Examples are XMLServlet (Java Servlet to convert XML to HTML on the fly) from Cerium Component Software, EDI/XML Authoring Tool from Redix International and Dynatag (converts word processing documents into DynaText electronic books) from Inso.

Parsers/Processors

XML parsers process an XML document and make it available to an application as an interface that the application understands. Standard interfaces for this are SAX and DOM (described in the next section). Thus the purpose of an XML parser is to take the XML document text input and convert it into objects that can be used by the corresponding programming language. XML parsers can be validating (check conformance of an XML document against a DTD) or nonvalidating (check only well-formatted XML documents). Examples of XML parsers are IBM's XML4C (for C++) and XML4J (for Java), Oracle's XML Parser for Java and Microsoft's XML Parser.

Tools/Utilities

Several tools and utilities for XML that provide additional processing and services that enhance XML processors are coming on the market. Since the tool supplied by a particular vendor provides unique functionality, it's hard to pick examples here. However, the examples on my list are SAXON (a Java interface for processing XML documents) from Michael Kay, Java Project X package from Sun Microsystems and xWingML (builds XML documents that define the complete Java Swing GUI) from Bluestone.

Soft Wired

www.java-messaging.com/ibus

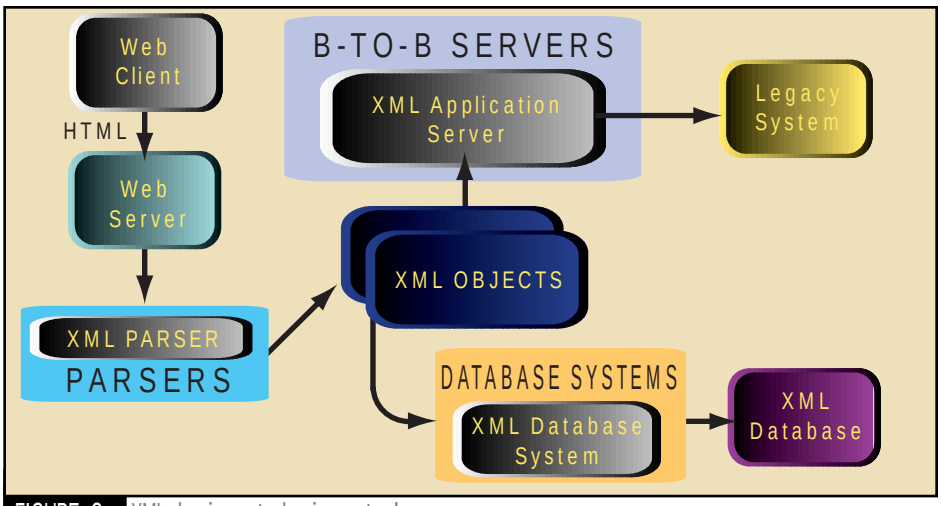


FIGURE 2: XML business-to-business tools

Database Systems

As mentioned earlier, XML basically defines data formatting. Databases define data storage and retrieval mechanisms. It's therefore not surprising that database vendors are jumping on the XML bandwagon. This includes RDBMS as well as OODBMS databases. The databases provide persistent stores for XML documents. Examples of database system products are eXcelon from Object Design and Oracle8*i* from Oracle.

Business-to-Business Servers

In my opinion, of all the product categories described in this article, this category is the one in which the heavyweight products will show up. XML is all about data formatting, exchange and translation between applications. The XML server product category is one in which businesses that have bet the farm on XML are staking their claim. Business-to-business XML integration servers connect applications to existing Web sites and provide the ability to leverage Web protocols in integrating business applications directly over the Web to existing legacy applications. The products in this category include tools for transaction processing and object serialization tools for application-to-application information exchange using XML. The types of applications integrated by these application servers include e-commerce applications, EDI, supply chain integration and thin-client enabled devices, desktop applications, and so on. Bluestone's XML-Server and WebMethods' B2B Server are examples of products in this category.

XML Content Management Tools

XML content management environments are used to build information exchange and translation systems from rich information repositories that combine the power of relational databases, query languages and search engines. Examples of products in this category are BladeRunner from InterLeaf and POET Content Management Suite from POET software.

Trading Places

As the editor of this "XML Focus" issue of ***JDJ***, I think that the market for XML is definitely getting there. The technologies are evolving so rapidly that it's hard to separate the hype from reality. As Sean Rhody, ***JDJ***'s editor-in-chief, said in his editorial "The XML Mambo" (***JDJ*** Vol. 4, issue 6): "Nice, but why is XML the next killer app?"

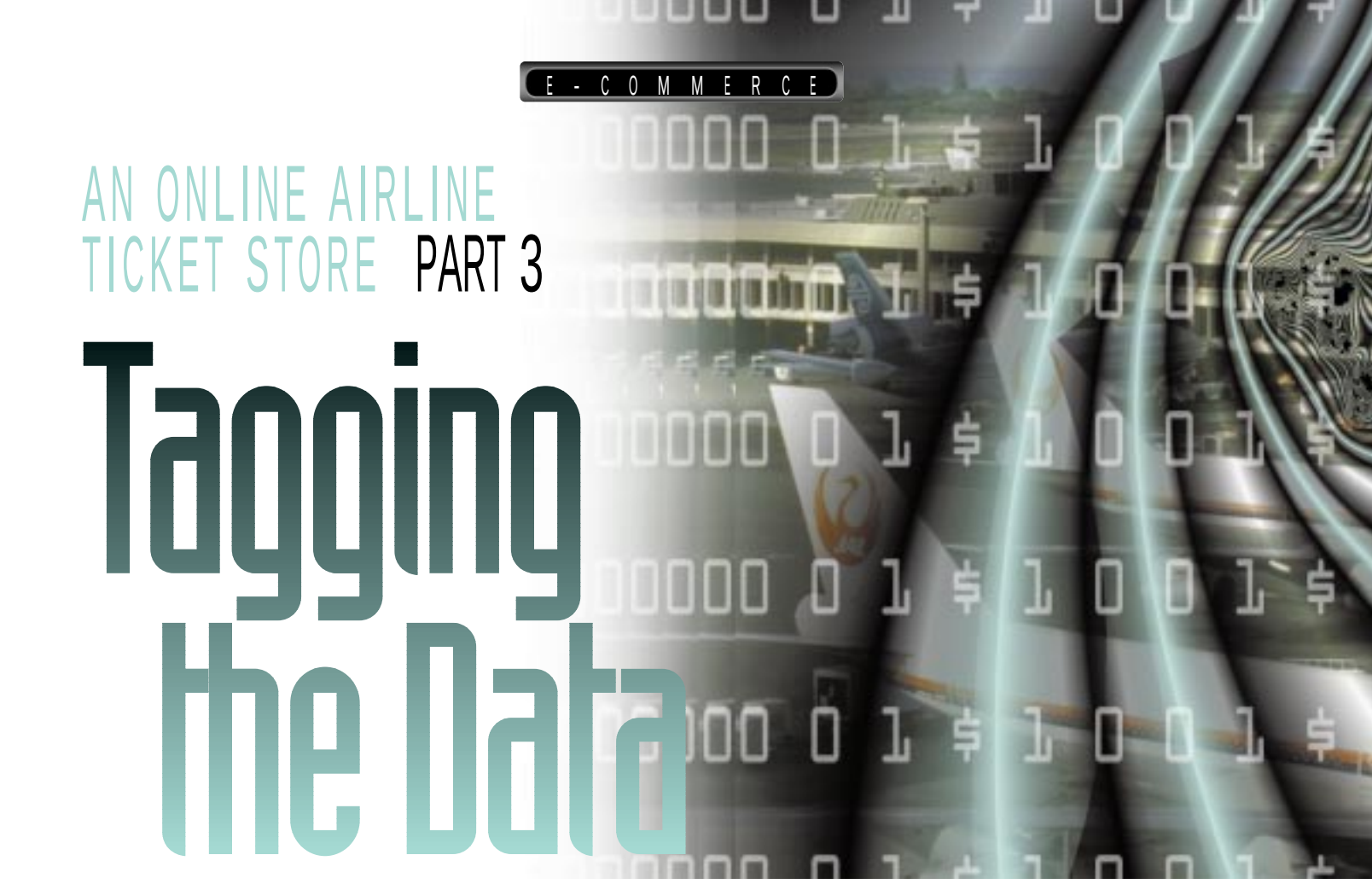
These are the kinds of questions SYS-CON would like to start addressing. In fact, *is* XML the next killer app? Or is it just an enabler for the next wave of killer apps? Most of you who are starting to get involved in XML are probably trying to figure out where to start and what XML is going to do for you.

We'd like to help you by directing our writers to address specific issues in the XML industry. We encourage readers, writers and vendors to provide feedback to us by filling out the online XML survey form at JavaDevelopersJournal.com regarding the possibility of an XML publication.

Ajit_Sagar@i2.com

OASIS

www.oasis-open.org



AN ONLINE AIRLINE TICKET STORE PART 3

Tagging the Data

The first part of this series appeared in **JDJ** *June and the second part appeared in* **JDJ** *July*

WRITTEN BY AJIT SAGAR

This is the third in a series of articles focused on using Java and ColdFusion technologies to develop an Online Ticket Store application. In the July issue of ***JDJ*** we went through the ticket reservation system for our online store. We took a look at how the actual protocol used for communicating with the airline back offices could be abstracted at the Service Access tier.

This month ***JDJ*** is focusing on XML, which brings us to an aspect of our store transactions that we haven't paid much attention to – data formatting. Let's pause to think about the type of data we're transporting across the different tiers of our architecture. Primarily, the end user submits his or her search criteria for an airline ticket and gets back a response from the airline back office. During this transaction the data goes through several tiers of a distributed application. Another part of our online store is the module that offers goods for purchase or lease. (This will be developed in the next article in this series.) The data transferred in the airline store's purchase and lease transactions will undergo a route similar to the data for the ticket reservations.

To make this application scalable to different storefronts, we'll need a standard data format to represent the transaction objects, such as an airline ticket query and the corresponding confirmation. Indeed, this kind of format is one of the main reasons for XML's existence.

I'll begin with a definition of the data objects for the Online Ticket Store, which will include objects that will be used in the next article for selling and leasing goods offered in the store. Such objects include clothes and souvenirs as well as portable CD players, laptops, CDs and books that can be leased for the duration of the flight. In this article we'll encapsulate the string arguments we pass into the TicketBrokerServlet and the StoreServlet (defined in next month's article) into XML structures. The servlets will parse the XML structures into Java objects for processing in the middle and back-office tiers.

This article focuses only on data formatting for data objects using XML. As I'm assuming that readers are familiar with XML structures and parsing, I won't cover the basics here. Readers will need a basic familiarity with Java servlets to follow the discussion, but those interested in the application itself who don't care much about XML can skip forward to the next part in the series without missing anything. In the next article we'll discuss the online store part of the application, i.e., the merchandise buying and leasing operations offered by the store.

Data Interchange Formats

Our application consists of the airline reservation system and the online store. The data objects for the airline reservation system are:

- ***Ticket query***
- ***Ticket quote***
- ***Booking request***
- ***Booking confirmation***

Although the online store won't be described until the next issue, I'm going to define the objects for it now:

- ***Lease order***
- ***Lease confirmation***
- ***Purchase order***
- ***Purchase confirmation***

Note that in the above data objects there are three types of confirmations. To keep things simple, let's limit our data definitions to one confirmation structure with an attribute that indicates which operation is being confirmed.

The end-to-end interaction in our distributed application starts from the Client UI tier and ends in the Application Services tier. Figure 1

shows a breakdown of this data interaction. The data from the Client UI tier to the Merchant Server tier is in HTML, since our front end to the Merchant Server is a page served up by Allaire's ColdFusion (please see the corresponding issues of ***ColdFusion Developer's Journal***, Vol. 1, issues 5 and 6). The end-user client UI can run in a browser; hence we get a virtual Internet storefront. The data representation for the interaction between the Service Access tier and the Application Services tier is going to depend on the protocol. With RMI, serialized Java objects are passed across the wire. With CORBA, CORBA structures constitute the format for the data interchange. With raw sockets and Java servlets, data is exchanged using serialized data streams.

This leaves the data interchange between the Merchant Server and the Service Access tiers. If this application were migrated to the real world, our online store would have to communicate with several airlines and airport stores. In such cases defining the data objects, e.g., Purchase Order, becomes a nightmare if each back office has its own definition of the object. Luckily, standard data formats for most industry verticals are emerging in the e-business market. The most prominent of these is XML, which holds the promise of spanning all of e-business. Similarly, if our Service Access tier were transported to another application and the data formats were different, it would be a substantial task to update our data structures to the new formats.

Data Objects for Reserving and Booking Tickets

The attributes of the ticket query and response from the airline back-office tier were defined in ***JDJ***'s July issue. The fields in the ticket query are shown below. Example values are assigned to the fields:

DEPARTURE_CITY = ***"Smallville"***
ARRIVAL_CITY = ***"Gotham"***
BEGIN_DATE = ***"31/01/1999"***
END_DATE = ***"01/01/2000"***
NUMBER = ***"2"***
CLASS = ***"First"***
PREFERENCE = ***"Window"***
SMOKING = ***"No"***

The result of this query is returned as a ticket quote from the various airline back offices to the Service Access tier, which creates a best quote and returns it to the Merchant Server tier. This quote has the following fields:

REFERENCE_NO = ***"SA2345678"***
AIRLINE = ***"SeemaAir"***
DEPARTURE_CITY = ***"Smallville"***
DEPARTURE_DATE = ***"31/01/1999"***
DEPARTURE_TIME = ***"10:00 P.M."***
ARRIVAL_CITY = ***"New York"***
ARRIVAL_DATE = ***"01/01/2000"***
ARRIVAL_TIME = ***"1:30 A.M."***
NO_OF_SEATS = ***"2"***
CLASS = ***"First"***
SEATING_PREFERENCE = ***"Window"***
SMOKING = ***"No"***
PRICE = ***"$1234.56"***

The data structure for a booking request has the following fields:

NAME = ***"Clark Kent, Lois Lane"***
REFERENCE_NO = ***"SA2345678"***
AIRLINE = ***"SeemaAir"***

A basic assumption here is that the reference number will be unique for an airline and the back office can access the details of the flight based on the confirmation number. Based on this input, the airline will send back a confirmation object with just one field:

CONFIRMATION_NO = ***"SA2345678"***

The hierarchy of the XML documents in DOM for the ticket reservation operations is shown in Figure 2. The XML file is shown in Listings 1–4. Listing 4 is the XML file for a confirmation. The <Confirmation> tag has one attribute, "Type," which indicates the type of confirmation. Our broker can return confirmations of the type "Ticket," "Lease" and "Purchase." I'm providing only an example of the ticket reservation confirmation. The confirmations for leasing and purchasing merchandise will be similar.

Data Objects for Selling and Leasing Goods

Let's go ahead and define the attributes of goods sold in the store. An item that may be offered by the store will have the following fields (example values are assigned to the fields):

ITEM_NAME = ***"CD Player"***
ITEM_ID = ***"cd30056"***
QUANTITY = ***"2"***

An order for the purchase of an item (or items) will also need information about the person making the purchase. This will consist of the following fields:

NAME = ***"Clark Kent"***
ADDRESS = ***"123 Tiny Lane, Smallville, Kansas, 12345"***

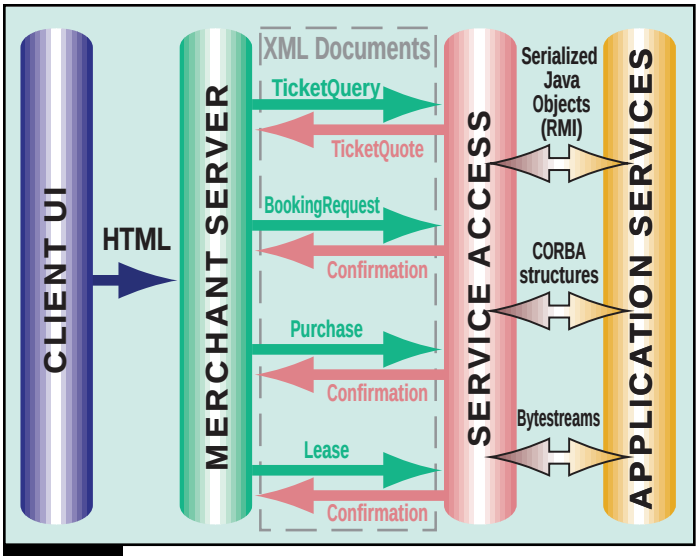


FIGURE 1: Data interchange between online ticket store tiers

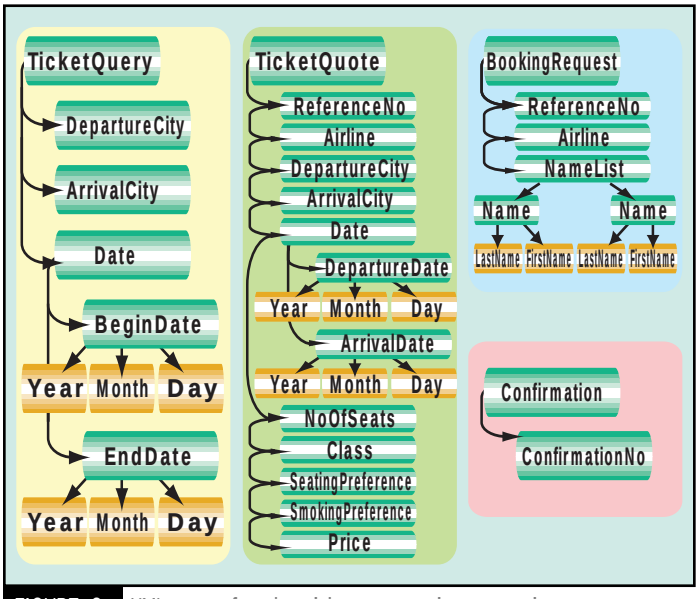


FIGURE 2: XML trees for the ticket reservation operations

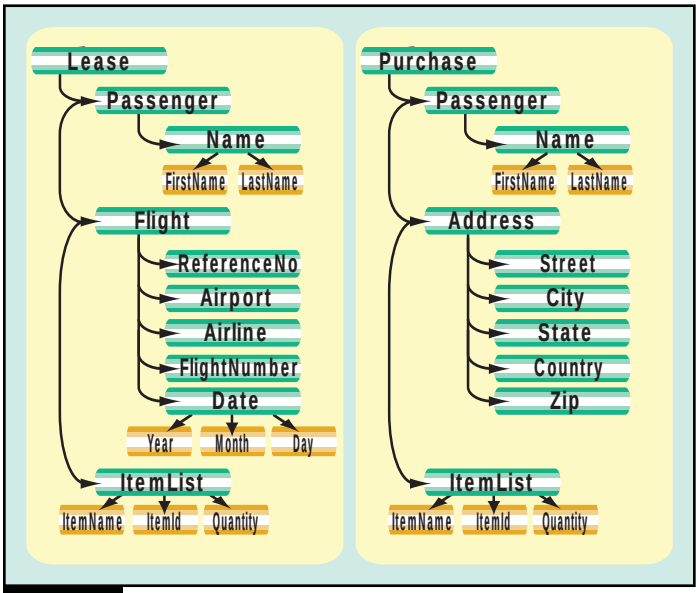


FIGURE 3: XML trees for the purchase and lease operations

The fields listed above are common for both purchase and lease options. Some fields, however, are characteristic of the lease operation because the lease is going to be linked to the flight the person takes (the equipment is leased for the duration of the flight and hence must be available at the airport). So the additional fields for the lease are:

AIRPORT = *"Sunita International Airport"*
AIRLINE = *"SeemaAir"*
FLIGHT_NO = *"SA 123"*
DEPARTURE_DATE = *"12/01/2000"*

The hierarchy of the XML documents in DOM for the buy/lease operations is shown in Figure 3. The XML file is shown in Listings 5 and 6.

Back to Basics: HTTP GET and HTTP POST

Now that we have our data object definitions, what should we do with them? To actually use the XML data structures, we need to have modules in the Merchant Server tier to send the data objects to the Service Access tier. We also need modules in the Service Access tier to parse the XML data into Java objects and then send them to the appropriate back-office tier for processing. In the case of the Merchant Server tier, the data is actually submitted via a ColdFusion template. The Service Access tier receives all requests via a servlet. The data for ticket reservations is processed by the TicketBrokerServlet (as described in the July article). The data for the merchandise purchase and leasing will be processed by the StoreServlet (described in the next article). Here we'll look at the code that parses the XML text.

So far, we've used the GET method to send data to the TicketBrokerServlet; the ticket request is sent as a parameterized query with the URL string to the server. The POST method is more flexible, however, because there are no limitations about parameter format and length. When transmitting XML structures, the length of the query string can be substantial. Some servers may limit the query string of the GET parameters to 240 characters, and the method shouldn't be used to send information to the server. Its main purpose is to read information from the server.

Hence our Java servlets need to pick up POST requests sent by the "client." In a servlet this is done by the doPost() method. Let's look at the methods in the TicketBrokerServlet that can pick up the POST request and parse the XML content into Java objects. For each XML data structure we'll need a corresponding Java object (shown in Listing 7). All the Java data objects are given in one listing to conserve space. Each data object is represented as a simple Java class with one full-argument constructor and getter methods. Typically, each object will have its own listing. As you can see, the complete code isn't given in this article; most of the objects are stubbed out. Instead, I've provided the complete code for the TicketQuery and TicketQuote objects to illustrate the XML parsing and document creation. Please refer to ***JDJ***'s Web site at www.sys-con.com/java for the complete code listing. The code for the TicketBrokerServlet that handles the TicketQuery and TicketQuote is shown in Listing 8. The listing for the merchandise purchase and leasing (not given here) will be available at the Web site.

IBM's XML Parser for Java is used for parsing the XML document that's sent via the HTTP POST request. Let's look at the doPost() method of the TicketBrokerServlet in Listing 8. (The complete code for this servlet isn't discussed here and is omitted from the listings due to space constraints. This code was discussed in the previous articles. Where the code is stubbed out, I have inserted the comments "// etc.")

TicketBrokerServlet

The request and response streams are copied into request_ and response_ variables for future use. Next, the method parseParameters() is called. This method parses the input parameters and creates the appropriate Java object. The XML processing for this application is done in the XMLProcessor class (Listing 9). The TicketBrokerServlet first creates a new instance of the XMLProcessor object. It then calls the initParser() method on the TicketStoreXML class. The servlet's input stream is passed in as a parameter for the parser to parse the XML document that the servlet received in its input stream.

Next, the method getQueryType() is called on the XMLProcessor reference (xp) to check what kind of an operation was invoked on the TicketBrokerServlet. If the type is a "TicketQuery," the method processTicketQuery() is called, which returns a TicketQuery object from the XML document. Following this, the local method processQuery() is called. This method goes across to the Application Services tier and gets back a TicketQuote.

The method processQuery() was discussed in the previous article. I have stubbed out the functionality to keep the listing short and focused. To test the code, a TicketQuote is created directly. In the application a call to getQuotes() will be made to the clientManager_; this will return a vector of quotes, and the local getBestQuote() method would be called to obtain the best quote. The end result is the same. We get an object that encapsulates the best quote from the back-office tiers.

The next call made on the xp is processTicketQuote(). This takes in the TicketQuote object we just obtained and returns a String, which is the corresponding XML document that needs to be sent back to the Merchant Server tier. This is appended to the result_ string and the string is written back via the servlet's output stream, toClient_.

XMLProcessor

Finally, let's look at the class that does all the XML work. Note that this class imports classes from the packages com.ibm.xml parser and org.w3c.dom. The former contains the classes for IBM's XML for Java parser. The latter contains the W3C classes for XML documents.

The method initParser() creates a new instance of the XMLParser. Next, the data is read in from the InputStream, input, which is actually the TicketBrokerServlet's input stream. Finally, the root of the document is initialized.

The method processTicketQuery() is called from the TicketBrokerServlet's processQuery() method. This method extracts the data from the XML document received via the POST method and uses it to construct a TicketQuery object. It returns the TicketQuery object to the calling method. The first few lines of the method call the local method processSingleTag(), a utility method for processing a single tag in an XML document. Next, the Date element is parsed. The begin and end dates are created as strings beginDate and endDate. Once the dates are parsed, the data is used to create and return a new TicketQuery object to the TicketBrokerServlet.

The next two methods are utility methods that aid in parsing XML documents. The processSingleTag() method is a utility method that takes in a tag String and an XML Element and returns its text value. A NodeIterator is created and the first element in the list is extracted from the Element that was passed in as an argument. Finally, the text value of this extracted element is passed back. The processListTag() method is similar, except that the code iterates through a list of elements and passes back a Vector of Elements to the calling method. The NodeIterator class is used for this purpose.

The last method in this class is used to create an XML document from a TicketQuote object, which is passed in as an argument to the method. A new TXDocument is created and the root element ("TicketQuote") is created and inserted into the document. Next, the version for XML is set. This is followed by a declaration of variables representing various levels of the XML document structure. The XML document is going to be three levels deep so three TXElements are created. The rest of the method creates and adds the elements to the tree. The values for the elements are obtained from the "quote" variable passed in as the argument to this method. The first element is the "ReferenceNo." The "child" variable is initialized to a new "referenceNo" TXElement object. Next, the text value for this element is obtained from the quote object by calling getReferenceNo(). This is added to the child element, which in turn is added to the root element.

The rest of the method adds other elements in a similar fashion. In the case of the Date elements, the tree is three levels deep. Hence the grandChild and greatGrandChild variable are used. At the end of the method, the document is written to a StringWriter variable and returned to the calling method.

Conclusion

We've now been through the entire exercise of creating XML documents for transporting data from one tier of a distributed application to another, converting that XML document into a language-specific object, sending it across to another tier, getting back another object, converting this new object to an XML document and transporting it back to the original tier. What have we gained? We've certainly added a lot of overhead by using XML tags instead of plain strings.

Well, the big gain is in standardization and clarity in data formatting. In the next couple of years, if all goes well, XML standards will start solidifying in the marketplace for several industry verticals. These standards will help provide universal definitions of data structures such as a "Reservation" or an "Order." Indeed, such standards are already appearing in today's marketplace.

As for the overhead, that's a price we'll have to pay for the benefits of standardization. Take TCP/IP, for instance. When I first looked at a TCP packet, I was shocked to discover that for each character of the login I sent across the wire, the stack added 53 characters that encapsulated the data in the packet. However, think of the universality that standards such as TCP have brought to the computing world.

Author Bio

Ajit Sagar, a member of the technical staff at i2 Techologies in Dallas, Texas, holds an MS in computer science and a BS in electrical engineering. He focuses on Web-based e-commerce applications and architectures. Ajit is a Sun-certified Java programmer with nine years of programming experience, including two and a half in Java.

Ajit_Sagar@i2.com

Listing 1: TicketQuery XML Structure

```
<?xml version="1.0"?>
<TicketQuery>
 <DepartureCity>Smallville</DepartureCity>
  <ArrivalCity>Gotham</ArrivalCity>
   <Date>
    <BeginDate>
     <Year>1999</Year>
     <Month>12</Month>
     <Day>31</Day>
     </BeginDate>
     <EndDate>
      <Year>2000</Year>
      <Month>1</Month>
      <Day>1</Day>
      </EndDate>
 </Date>
 <NoOfSeats>2</NoOfSeats>
 <Class>First</Class>
 <SeatingPreference>Window</SeatingPreference>
 <SmokingPreference>No</SmokingPreference>
</TicketQuery>
```

Listing 2: TicketQuote XML Structure

```
<?xml version="1.0"?>
<TicketQuote>
 <ReferenceNo>"SA2345678"</ReferenceNo>
 <Airline>SeemaAir</Airline>
 <DepartureCity>Smallville</DepartureCity>
   <ArrivalCity>Gotham</ArrivalCity>
    <Date>
         <DepartureDate>
              <Year>1999</Year>
              <Month>12</Month>
              <Day>31</Day>
         </DepartureDate>
         <DepartureTime>
              <Hour>22</Hour>
              <Minutes>00</Minutes>
              </DepartureTime>
         <ArrivalDate>
              <Year>2000</Year>
              <Month>1</Month>
              <Day>1</Day>
         </ArrivalDate>
         <ArrivalTime>
              <Hour>1</Hour>
              <Minutes>30</Minutes>
         </ArrivalTime>
 </Date>
 <NoOfSeats>2</NoOfSeats>
 <Class>First</Class>
 <SeatingPreference>Window</SeatingPreference>
 <SmokingPreference>No</SmokingPreference>
 <Price>"$1234.56"</Price>
</TicketQuote>
```

Listing 3: BookingRequest XML Structure

```
<?xml version="1.0"?>
<BookingRequest>
     <ReferenceNo>"SA2345678"</ReferenceNo>
     <Airline>SeemaAir</Airline>
     <NameList>
          <Name>
```

```
            <LastName>"Kent"</LastName>
            <FirstName>"Clark"</FirstName>
        </Name>
        <Name>
            <LastName>"Lane"</LastName>
            <FirstName>"Lois"</FirstName>
        </Name>
    /NameList>
</BookingRequest>
```

Listing 4: Confirmation XML Structure

```
<?xml version="1.0"?>
<Confirmation Type="Ticket">
 <ConfirmationNo>"SA2345678"</ConfirmationNo>
</Confirmation>
```

Listing 5: Lease XML Structure

```
<?xml version="1.0"?>
<Lease>
 <Passenger>
   <Name>
     <LastName>"Kent"</LastName>
     <FirstName>"Clark"</FirstName>
   </Name>
 </Passenger>
 <Flight>
  <ReferenceNo>"SA2345678"</ReferenceNo>
   <Airport>"Sunita International Air-
    port"</Airport>
     <Airline>"SeemaAir"</Airline>
     <FlightNumber>"SA 123"</FlightNumber>
        <Date>
           <Year>1999</Year>
           <Month>12</Month>
           <Day>31</Day>
        </Date>
    </Flight>
    <ItemList>
        <ItemName>"CD Player"</ItemName>
        <ItemId>cd00321</ItemId>
        <Quantity>1</Quantity>
        <ItemName>"Book"</ItemName>
        <ItemId>bookx453</ItemId>
        <Quantity>1</Quantity>
    </ItemList>
</Lease>
```

Listing 6: Purchase XML Structure

```
<?xml version="1.0"?>
<Purchase>
    <Name>
        <LastName>"Kent"</LastName>
        <FirstName>"Clark"</FirstName>
    </Name>
    <Address>
        <Street>"123 Tiny Lane"</Street>
        <City>"Smallville"</City>
        <State>"Kansas"</State>
        <Country>USA</Country>
        <ZIP>12345</ZIP>
    </Address>
    <ItemList>
        <ItemName>"CD Player"</ItemName>
        <ItemId>cd00321</ItemId>
        <Quantity>1</Quantity>
        <ItemName>"Book"</ItemName>
        <ItemId>bookx453</ItemId>
        <Quantity>1</Quantity>
    </ItemList>
</Purchase>
```

Listing 7: Data Objects for the Online store

```
// Please separate the listings into indi-
vidual
// files: e.g., TicketQuery.java, Tick-
etQuote.java

import java.io.*;
import java.text.*;
import java.util.*;


// ****************************************
// ** file TicketQuery.java
// ****************************************

public class TicketQuery implements Serial-
izable {

  DateFormat df_ =

DateFormat.getDateInstance(DateFormat.SHORT
);

  // Fields for a ticket query
  private long queryId_ = 0;
  private String departureCity_ = null;
  private String arrivalCity_ = null;
  private Date beginDate_ = null;
  private Date endDate_ = null;
  private int noOfSeats_ = 0;
  private String seatClass_ = null;
  private boolean aisle_ = false;
  private boolean smoking_ = false;

  // Full-arg constructor
  public TicketQuery (String departureCity,
                      String arrivalCity,
                      String beginDate,
                      String endDate,
                      String noOfSeats,
                      String seatClass,
                      String aisle,
                      String smoking) {

    // Generate new query ID
    queryId_ = generateQueryId();

    departureCity_ = departureCity;
    arrivalCity_ = arrivalCity;

    // The date Strings need to be parsed
    setBeginDate(beginDate);
    setEndDate(endDate);
    setNoOfSeats(noOfSeats);
    setSeatClass(seatClass);
    setAisle(aisle);
    setSmoking(smoking);
  }

  public long generateQueryId () {
     // The current time is used to gener-
     // ate the queryId.
     // This way, we get a time stamp for
     // the query.
     return System.currentTimeMillis();
  }

  // -- Setter methods are required for
  //    these fields

  public void setBeginDate (String begin
  Date) {
      try {
          beginDate_ = df_.parse(begin
          Date);
      }
      catch (ParseException pe) {
        pe.printStackTrace();
      }
  }

  public void setEndDate (String endDate)
{
      try {
          endDate_ = df_.parse(endDate);
      }
      catch (ParseException pe) {
          pe.printStackTrace();
      }
  }

  public void setNoOfSeats(String noOf-
Seats) {
      noOfSeats_ = Integer.parseInt(noOf
      Seats);
  }

  public void setSeatClass (String seat-
  Class) {
      seatClass_ = seatClass;
  }

  public void setAisle (String aisle) {
      if (aisle.equalsIgnoreCase("YES"))
          aisle_ = true;
      else
          aisle_ = false;
  }

  public void setSmoking(String smoking) {
      if (smoking.equalsIgnoreCase("YES"))
          smoking_ = true;
      else
          smoking_ = false;
  }

  // -- Getter methods for all the fields.

  public long getQueryId ()
      { return queryId_; }
  public String getDepartureCity ()
      { return departureCity_; }
  public String getArrivalCity ()
      { return arrivalCity_; }
  public Date getBeginDate ()
      { return beginDate_; }
  public Date getEndDate ()
      { return endDate_; }
  public int getNoOfSeats ()
      { return noOfSeats_; }
  public String getSeatClass ()
      { return seatClass_; }
  public boolean isAisle ()
      { return aisle_; }
  public boolean isSmoking ()
      { return smoking_; }
}

// ****************************************
// ** file TicketQuote.java
// ****************************************

public class TicketQuote implements Serial-
izable {

  // Fields for a ticket query
  private String referenceNo_ = null;
  private String airline_ = null;
  private String departureCity_ = null;
  private String arrivalCity_ = null;

  // The Date fields include the time val
  // ues
  private Date departureDate_ = null;
  private Date arrivalDate_ = null;

  private int noOfSeats_ = 0;
  private String seatClass_ = null;
  private boolean aisle_ = false;
  private boolean smoking_ = false;
  private String price_ = "$0.00";

  // Full-arg constructor
  public TicketQuote (String referenceNo,
                      String airline,
                      String depar-
tureCity,
                      String arrivalCity,
                      Date departureDate,
                      Date arrivalDate,
                      int noOfSeats,
                      String seatClass,
                      boolean aisle,
                      boolean smoking,
                      String price) {

    referenceNo_ = referenceNo;
    airline_ = airline;
    departureCity_ = departureCity;
    arrivalCity_ = arrivalCity;
    departureDate_ = departureDate;
    arrivalDate_ = arrivalDate;
    noOfSeats_ = noOfSeats;
    seatClass_ = seatClass;
```

```
    aisle_ = aisle;
    smoking_ = smoking;
    price_ = price;
  }

  // -- Getter methods for all the fields.

  public String getReferenceNo ()
      { return referenceNo_; }
  public String getAirline ()
      { return airline_; }
  public String getDepartureCity ()
      { return departureCity_; }
  public String getArrivalCity ()
      { return arrivalCity_; }
  public Date getDepartureDate ()
      { return departureDate_; }
  public Date getArrivalDate ()
      { return arrivalDate_; }
  public int getNoOfSeats ()
      { return noOfSeats_; }
  public String getSeatClass ()
      { return seatClass_; }
  public boolean isAisle ()
      { return aisle_; }
  public boolean isSmoking ()
      { return smoking_; }
  public String getPrice () { return
price_; }
}


  // PLEASE GO TO www.sys-con.com/java
     // FOR THE FOLLOWING CLASSES

class Name {}
class BookingRequest {}
class Confirmation {}
class Address {}
class Item {}
class Lease {}
class Purchase {}
```

Listing 8: TicketBrokerServlet.java

```
import java.io.*;
import java.util.*;
import java.rmi.*;
import java.text.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class TicketBrokerServlet extends
HttpServlet {
  protected HttpServletRequest request_ =
  null;
  protected HttpServletResponse response_
  = null;
  protected StringBuffer result_ = null;

  XMLProcessor xp_ = null;
  PrintWriter toClient_ = null;
  DateFormat df = DateFormat.getDateIn-
  stance(DateFormat.SHORT);

  // TicketClientManager clientManager_ =
  // null;


  public void doPost (HttpServletRequest
  request,
              HttpServletResponse response)
    throws ServletException, IOException {
    try {

      request_ = request;
      response_ = response;

      // Get a reference to the output
      // stream
      toClient_ = new
PrintWriter(response_.getOutputStream());

      if (toClient_ == null) {
        response_.sendError(1, "Null
toClient_ stream");
      }

      // etc.

      result_ = new StringBuffer();
      processParameters();


toClient_.println(result_.toString());
      toClient_.flush();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }

  private void processParameters () {
    // XMLProcessor for converting XML to
    // Java
    xp_ = new XMLProcessor();

    try {
      InputStream in = request_.getInput-
      Stream();
      xp_.initParser(request_.getInput-
      Stream());
    }
    catch (IOException ioe) {
      ioe.printStackTrace();
    }

    String queryType = xp_.getQueryType();
    if (queryType.equals("TicketQuery")) {
        TicketQuery ticketQuery =
        xp_.processTicketQuery();
        processQuery(ticketQuery);
    }
    else if (queryType.equals("BookingRe-
    quest")) {
        // BookingRequest bookingRequest =
        // xp_.processTicketBooking();
        // processBooking(bookingRequest);
    }
  }

  private void processQuery (TicketQuery
  query) {
    Vector quotes = null;

    // !! Stubbed code. Typically, this
    // will be replaced by a call
    // !! to the appropriate client via
    // the TicketClientManager.

    // etc.

    try {
      TicketQuote ticketQuote =
          new TicketQuote("SA123456",
          "SeemaAir",
          "SmallVille", "Gotham",

df.parse("12/31/1999"),

df.parse("1/1/2000"),
              2, "First", false, false,
              "$1234.56");

      String xmlQuote = xp_.processTick-
      etQuote(ticketQuote);
      result_.append(xmlQuote);
    }
    catch (ParseException pe) {
        pe.printStackTrace();
    }
  }

  //Process a booking request.
  private void processBooking (BookingRe-
  quest bookingRequest) {
      // etc.
  }

  // Return best quote from the quotes
  // obtained from
  // the Application Services tier
  private TicketQuote getBestQuote (Vector
  quotes) {
      // etc.
      return null;
  }
```

Listing 9: java: XML processor for the Online Ticket Store

```
import com.ibm.xml.parser.*;
import org.w3c.dom.*;
import java.io.*;
import java.util.*;
import java.text.*;

public class XMLProcessor {
  Document doc_;
  TXElement root_;
  PrintWriter pw_ = null;

  Calendar cal_ = Calendar.getInstance();

  // Initialize the XML parser and document
  public void initParser(InputStream input)
  {
    InputStream xmlStream =
    getXMLStream(input);

    Parser p = new Parser("XMLParser");
    doc_ = p.readStream(xmlStream);
    root_ = (TXElement) doc_.getDocu-
    mentElement();
  }

  // Get a stream with the XML content only
  public InputStream getXMLStream (Input-
  Stream input) {

    byte[] array = new byte[10000];
    String str = null;
    try {
        input.read(array);
        str = new String(array);

        // Strip the null characters
        int i = str.indexOf('\0');
        str = str.substring(0, i);
    }
    catch (IOException e) {
        e.printStackTrace();
    }

    int beginXML = str.indexOf("<?xml ver-
    sion");
    int endXML = str.lastIndexOf(">") + 1;
    str = str.substring(beginXML, endXML);
    return new
ByteArrayInputStream(str.getBytes());
  }

  // Root tag name
  public String getQueryType () {
      return root_.getName();
  }

  // Extract XML into a Java TicketQuery
  // object
  public TicketQuery processTicketQuery ()
  {
    String departureCity = processSingle-
    Tag("DepartureCity", root_);
    String arrivalCity =  processSingle-
    Tag("ArrivalCity", root_);
    String noOfSeats =
    processSingleTag("NoOfSeats", root_);
    String seatClass =
    processSingleTag("Class", root_);
    String seatingPref = processSingle-
    Tag("SeatingPreference", root_);
    String smokingPref = processSingle-
    Tag("SmokingPreference", root_);


    TXElement txe = null;
```

```
    NodeIterator iterator1 = root_.getEle-
    mentsByTagName("Date");

    // There is only one Date element
    TXElement dateElement =
    (TXElement)iterator1.toFirstNode();

    NodeIterator iterator2 =
    dateElement.getElementsByTagName("Begin
    Date");

    TXElement beginDateElement = (TXEle-
    ment)iterator2.toFirstNode();

    String beginYear =
    processSingleTag("Year", beginDateEle-
    ment);
    String beginMonth = processSingle-
    Tag("Month", beginDateElement);
    String beginDay =
    processSingleTag("Day", beginDateEle-
    ment);

    String beginDate = beginMonth + "/" +
    beginDay + "/" + beginYear;

    iterator2 = dateElement.getElementsBy-
    TagName("EndDate");

    TXElement endDateElement = (TXEle-
    ment)iterator2.toFirstNode();

    String endYear =
    processSingleTag("Year", endDateElement);
    String endMonth =
    processSingleTag("Month", endDateEle-
    ment);
    String endDay =
    processSingleTag("Day", endDateEle-
    ment);

    String endDate = endMonth + "/" + end-
    Day + "/" + endYear;

    return new TicketQuery(departureCity,
arrivalCity, beginDate, endDate,
                           noOfSeats,
seatClass, seatingPref, smokingPref);

  }

  // Process a single tag from an XML doc-
  // ument
  public String processSingleTag(String
tag, Element startNode) {
    TXElement e;
    Node n;
    String textValue = null;
    NodeIterator iter = startNode.getEle-
    mentsByTagName(tag);

    int length = iter.getLength();

    if (length != 1) {
      System.out.println("Houston we have
      a problem with tag: " + tag);
    }
    else {
      n = iter.toFirstNode();
      e = (TXElement) n;
      textValue = e.getText();
    }

    return textValue;
  }

  // Process a list of tags from an XML
  // document
  public Vector processListTag(String tag,
  Element startNode)
  {
    TXElement e, cElement, cTemp;
    Node n;
    Vector list = null;
    String text;

    System.out.println("tag = " + tag + ",
    startNode = " + startNode);
    NodeIterator iter = startNode.getEle-
    mentsByTagName(tag);

    int length = iter.getLength();

    if (length != 0) {
      list = new Vector();
    }

    for (int i = 0; i < length; i++) {
      n = iter.moveTo(i);
      e = (TXElement) n;
      list.addElement(e);
    }
    return list;
  }

  // Create an XML document from the Tick-
  // etQuote object
  public String processTicketQuote (Tick-
  etQuote quote)
  {
    TXDocument doc = new TXDocument();
    TXElement root = new TXElement("Tick-
    etQuote");
    doc.insert(root, 0);
    doc.setVersion("1.0");

    TXElement child, grandChild, great-
    GrandChild;
    TXText text;

    child = new TXElement("ReferenceNo");
    text = new TXText((quote.getReferen-
    ceNo()).toString());
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("Airline");
    text = new TXText((quote.getAir-
    line()).toString());
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("Depar-
tureCity");
    text = new TXText((quote.getDepar-
    tureCity()).toString());
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("ArrivalCity");
    text = new TXText((quote.getArrivalCi-
    ty()).toString());
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("Date");
    root.addElement(child);

    grandChild = new TXElement("Departure-
    Date");
    child.addElement(grandChild);

    Date date = quote.getDepartureDate();
    cal_.setTime(date);

    greatGrandChild = new
    TXElement("Year");
    text = new
    TXText(String.valueOf(cal_.get
    (Calendar.YEAR)));
    greatGrandChild.addTextElement(text);
    grandChild.addElement(greatGrandChild);

    greatGrandChild = new
    TXElement("Month");
    text = new
    TXText(String.valueOf(cal_.get
    (Calendar.MONTH)));
    greatGrandChild.addTextElement(text);
    grandChild.addElement(greatGrandChild);

    greatGrandChild = new
    TXElement("Day");
    text = new
    TXText(String.valueOf(cal_.get
    (Calendar.DAY_OF_MONTH)));
    greatGrandChild.addTextElement(text);
    grandChild.addElement(greatGrandChild);

    grandChild = new TXElement("Arrival
    Date");
    child.addElement(grandChild);

    date = quote.getArrivalDate();
    cal_.setTime(date);

    greatGrandChild = new
    TXElement("Year");
    text = new
    TXText(String.valueOf(cal_.get
    (Calendar.YEAR)));
    greatGrandChild.addTextElement(text);
    grandChild.addElement(greatGrandChild);

    greatGrandChild = new
    TXElement("Month");
    text = new TXText(String.valueOf
    (cal_.get(Calendar.MONTH)));
    greatGrandChild.addTextElement(text);
    grandChild.addElement(greatGrandChild);

    greatGrandChild = new
    TXElement("Day");
    text = new
    TXText(String.valueOf(cal_.get
    (Calendar.DAY_OF_MONTH)));
    greatGrandChild.addTextElement(text);
    grandChild.addElement(greatGrandChild);

    child = new TXElement("NoOfSeats");
    text = new TXText(String.valueOf
    (quote.getNoOfSeats()));
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("Class");
    text = new TXText((quote.getSeat-
    Class()).toString());
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("SeatingPrefer-
    ence");
    text = new TXText(String.valueOf
    (quote.isAisle()));
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("SmokingPrefer-
    ence");
    text = new TXText(String.valueOf
    (quote.isSmoking()));
    child.addTextElement(text);
    root.addElement(child);

    child = new TXElement("Price");
    text = new TXText((quote.getPrice())
    .toString());
    child.addTextElement(text);
    root.addElement(child);

    StringWriter dString = new String-
    Writer();
    doc.printWithFormat(new
    PrintWriter(dString));
    return dString.toString();
  }
}
```

▼▼▼▼ CODE LISTING ▼▼▼▼

Code listings for this article can also be located at www.JavaDevelopersJournal.com

CORBA and XML: Conflict or Cooperation?

These technologies won't replace each other — they're complementary

WRITTEN BY
ANDREW WATSON

very now and then the computer industry gets swept up in a wave of enthusiasm for some new silver bullet that's apparently going to solve everyone's problems overnight. Actually, these days the wild surges of millennial euphoria seem to come at annual intervals. Usually the technology in question is actually a step forward, able to solve real problems better or faster than was possible before. However, as word spreads about the power of the new technique, some people will inevitably try to apply it to the wrong problems.

It's a bit like the enthusiasm for microwave ovens when they first became cheap enough for anyone to buy – one could buy microwave cookbooks explaining how to use them to cook everything from a complete Christmas dinner to a soufflé. Fortunately, after a while sanity returned, and people now use microwaves for what they're best at, and have gone back to making toast in the toaster or roasting the turkey in the oven, just as they always did, because they're the best tools for the job.

The same is true in the computer business, and as with cooking gadgets, it's important to get the balance right. Pointing out that you shouldn't try to make soup in your breadmaker doesn't in anyway diminish the fact that it's very, very good at making bread. In the same way, this article aims to put the current enthusiasm for XML in perspective without in any way detracting from or criticizing XML, which is an excellent tool for the job for which it was designed. However, the question "Will XML replace middleware?" is being asked so often at the moment that it seems appropriate to pen a few words on what applications XML is (and is not) suited for, and in particular why it isn't going to replace middleware solutions like CORBA (or vice versa, for that matter). To do this properly, we have to start with a little history. So, are you sitting comfortably? Then we'll begin.

A Little History

XML – eXtensible Markup Language – is a simplified subset of a previous markup language standard called SGML (Standard Generalized Markup Language) and was devised by a committee of the World Wide Web consortium in response to the need for a generalization of HTML, the HyperText Markup Language used to format Web pages.

SGML was conceived as a successor to document-markup languages like TeX, troff and nroff. These languages add formatting directives to plain text to tell typesetters, laser printers and other high-quality output devices how to format the text in various fonts of different sizes and styles. When they first appeared in the 1960s, markup languages were designed to be written by hand; one would use a text editor to create a plain text document, adding in the occasional markup directive to indicate that some piece of text should be printed in bold or centered or whatever. Of course, it was important to make sure there was no confusion between the content and the markup directives, so each family of markup languages had a set of conventions for separating them. For instance, in nroff and troff the directives are on lines beginning with a full stop (or period), while TeX begins directives with a "\" character.

As the use of markup languages became widespread, macros were added as a convenience feature. If headings in your document are to be displayed in centered bold 14 point Helvetica, it would soon get tedious to write four directives to change font, size, weight and justification for each heading. With a macro facility one can define a single command to do all this. Better yet, if you later decide your headings should be in Zapf Chancery instead, changing the definition of the "heading" macro automatically does the job everywhere you've used the macro.

Structure vs Presentation

Pretty soon authors creating complex documents found themselves maintaining large libraries of macro definitions and never using raw formatting directives in the documents at all. UNIX man pages are a good example – they're defined using the "man" macros for the nroff text formatter, making it easy to create manual pages with a consistent appearance.

During the '70s and '80s it became clear that the best way to use markup was by formalizing this approach: create a set of directives for describing the structure of the document as sections, subsections, bulleted items and so on, then separately define how to format those structural elements on paper. By keeping these two kinds of definitions (of structure and presentation) separate, altering the formatting of the documents or even reusing the content in new documents could be a completely mechanical process. Furthermore, automatic tools can process the documents to do jobs like building a contents page by listing all the headings. If your job is maintaining the many tons of paper documentation for (say) a commercial airliner, representing the logical structure of the document in this way is no small advantage since it allows the same source documents to be used to deliver information in a number of different formats. Again, UNIX man pages are a good example; when the manuals are printed on a high-resolution printer, using the same source text with a different library of (troff) macro definitions automatically creates book-quality manual pages rather than the screen-formatted pages generated from the same sources by nroff.

SGML, DSSSL and HTML

SGML was designed by ISO (the International Standards Organization) as a new standardized markup language that enshrined this separation of structure and presentation. To apply SGML one creates a Document Type Definition (DTD) that defines the set of valid tags for the documents being created, and uses DSSSL (the ISO-standardized Document Style Semantics and Specification Language that accompanies SGML) to define how to display text labeled with those tags. Between them the DTD and DSSSL definitions fill the same role as the macro library in older markup languages.

SGML has achieved limited success in large organizations that maintain very large documentation sets, but the SGML standard alone is over 500 pages, and the accompanying DSSSL (rhymes with "whistle") standard is also rather large and uses a syntax based on the Scheme programming language, which some people find hard to learn. Many users lack the will or resources to climb the SGML learning curve.

Meanwhile, at CERN in Geneva, Tim Berners-Lee was creating a simple SGML DTD to define a few document structure tags like "heading" and "numbered list" for defining the structure of documentation to be shared between nuclear physicists over computer networks. This simple application of SGML, called HTML, didn't have any accompanying way of defining the appearance of documents – that was provided by settings in the Web browser used to display the HTML document. The original HTML specification was simply a conforming SGML DTD describing the syntax of HTML documents, with the added wrinkle that one of the tags defined a way to hyperlink to another HTML document.

HTML, of course, has been much more widely used than SGML, but as its use spread, two problems became apparent. The first was that HTML defined only the structure of Web page elements, with no associated way of specifying their presentation, so the Web page designers had no way of controlling exactly how their creations looked. As Web pages became more sophisticated, with more graphic content, this became a serious problem, and ad hoc extensions were added to HTML to allow direct control of presentation by specifying fonts, font sizes, text colors and so on (which of course completely violates the original SGML design principles). At the same time, because the HTML had one fixed DTD, document designers had no way to create new structure tags to represent document structure in certain HTML applications. With neither an extension mechanism (like macros) nor a way of defining and controlling presentation, the original HTML fell neatly between two stools, and short-term product development pressures have inevitably pushed it toward being a presentation markup language that provides the Web-page designer with detailed control over how his/her document appears, rather than representing its logical structure. While this deals effectively with the primary purpose of Web pages, which is to be viewed by people using Web browsers, the increasing size and ubiquity of the Web is creating an increasing demand for Web pages that can be manipulated by Web-scanning "robots" such as the search engines that "read" and catalog millions of Web pages daily. It became clear that the lack of structured encoding threatened to slow down the development of the Web.

Enter XML

One solution to the problem of HTML's lack of structure would simply have been to step up one level and use SGML and DSSSL directly on the Web. However, the complexity of the ISO standards mitigated against this; something simpler was needed. In mid-1996 Jon Bosak, an influential member of the SGML community, persuaded W3C to set up an SGML Editorial Review Board and Working Group to define a simplified, extensible subset of SGML designed for the Web. The final XML 1.0 specification was published by W3C in February 1998, and will be complemented by two further specifications currently being prepared: XLL (the eXtensible Linking Language, for defining how XML documents are linked together) and XSL (the eXtensible Style Language, for defining how XML markup is formatted for display).

What Should XML Be Used For?

XML is being enthusiastically embraced in many application domains because a lot of applications need to store data intended for human use, but will also be useful to manipulate by machine. One example might be storing and displaying mailing list information. Defining and using an XML DTD for storing address data makes it comparatively easy to write applications to (say) generate address labels without inadvertently printing the phone number in the postcode field. There are a large number of initiatives to replace home-grown markup formats with applications of XML – examples include Bioinformatic Sequence Markup Language (BSML), Weather Observation Markup Format (OMF), the Extensible Log Format (XLF – a markup format for logging information generated by Web servers) and others for legal documents and real estate information, and many more. In each case the working group simply needs to define a DTD that defines the tags and how they can be legally combined. These DTDs can then be used with XML parsers and other XML tools to rapidly create applications to process and display the stored information in whatever way is required. Of course, there are still standardization issues to be addressed, such as who controls the libraries of tag definitions, how to manage version control in those libraries, and how to manage using multiple libraries simultaneously (especially when tag names collide). Nevertheless, using XML for these applications is a lot simpler than creating a completely new markup language from scratch every time, with a lot more scope for reusing the work of others.

One important point to note is that nowhere in the XML DTDs is there a way of specifying what an XML tag "means," just where it can be positioned in relationship to other tags, and (using XSL) how to format it on a display. Creators of XML DTDs naturally choose short descriptive names for their tags just as PC users usually choose short descriptive names for their files, so it's very appealing to think that XML files are "self-describing," because to an English speaker it's intuitive that an <address> tag labels an address or a <date-of-birth> tag labels a person's birthday. However, this is just the intuitive "meaning" we assign to the terms by assuming that the creator of the DTD used these words in the way we would expect; if the creator of the DTD had instead specified his tags in a foreign language or using some private code, we'd be none the wiser. XML files are in fact just as "self-describing" as a C program or a database schema.

What Shouldn't XML Be Used For?

The common thread in XML applications is that the document content is

intended to be read by people. Because XML is intended for marking up human-readable, textual data, it is by the same token a rather inefficient way of storing information that only needs to be machine-readable. The embedded XML tags provide a way to extract or format particular parts of the content, but the content itself won't usually be interpreted by the computers, only by the ultimate human user – which is why it makes sense to store it in human-readable form. Of course, it's perfectly possible to write parsers to read in (say) formatted floating-point numbers from an XML file so they can be processed, but it's relatively time-consuming, and the XML file would be relatively larger than one written in native floating-point format.

When the requirement is to exchange data between cooperating computer applications, there are other, more efficient ways of defining and storing the data. Traditionally these definitions of data formats for machine communication are called Interface Definition Languages (IDLs) because they're used for defining the interfaces between cooperating computer applications. In contrast to markup, which is used for the long-term storage of human-readable data, IDLs define the smaller packets of transient, machine-readable data that is exchanged between the components of a distributed application when some particular event occurs.

IDLs are the most visible components of a class of software known as "middleware," that class of software that is neither part of an operating system nor an application but is used to link the various parts of a distributed application spread across geographically separated computers. By their very nature, successful middleware solutions blend into the background, making few impositions on the users, designers and programmers of a distributed system. Today's most widely used middleware packages all implement the CORBA (Common Object Request Broker Architecture) specification, published by the OMG (Object Management Group).

Although IDL is the most visible aspect of middleware, there's much more to it than that: middleware solutions like CORBA also provide security to authenticate users and control access to resources, error handling to gracefully handle the failures inevitable in a distributed computing system, and a host of other support functions to keep computer networks running smoothly. In these sorts of distributed computing applications the data are transient, transferred between computers, often not permanently stored anywhere and probably never seen by human eyes. To use XML as the data encoding in such applications is less efficient than the compact, native machine representations used to marshal data in (for instance) the IIOP wire format used by CORBA implementations. Of course, if the requirement is to store data for the long term and extract human-readable summaries and reports, then XML would be the more appropriate medium – but for the data exchanges that tie together the components of a distributed system, using XML would be expensive and pointless.

Summary

XML and middleware are complementary technologies. XML is intended for the storage and manipulation of text making up human-readable documents like Web pages, while middleware solutions like CORBA tie together cooperating computer applications exchanging transient data that will probably never be directly read by anyone. Neither of these technologies will replace the other. Instead, they will increasingly be used together – not least in the specifications published by OMG, the body responsible for the CORBA specification.

Author Bio

Andrew Watson, OMG's VP and technical director, also chairs OMG's Architecture Board, which oversees the technical consistency of all OMG's specifications. Previously he chaired OMG's ORB task force, which was responsible for the development and deployment of the CORBA 2 specification. Before that he spent six years with the ANSA core team in Cambridge (UK) researching distributed object architectures, specializing in distributed object type systems.

andrew@omg.org

SYS-CON RADIO INTERVIEW

WITH BOB SUTOR, XML Industry Standards Manager, IBM, and Chief Strategy Officer, OASIS

Also participating in the interview was Ajit Sagar, guest editor of this XML Focus issue of *Java Developer's Journal*

JDJ: *Let's start off with XML in relation to IBM, Bob. Can you tell us why XML is important to the market and to IBM?*

Sutor: XML provides a standardized, flexible and powerful method for exchanging data among many different platforms and applications across the Internet. IBM is a leader in database and transaction systems, and XML will be the standard mechanism for interoperability among data warehouses, repositories and Web applications. Now IBM, of course, is a very big business, so in addition to the services and products we provide for our customers, we're using XML to make ourselves a first-class e-business. It's basically just a great way of representing data. I usually like to think about what XML is about using four "I"s:

I think of XML first of all being about I*nformation*, perhaps messages (exchanged between applications or platforms) or Web documents.

I*nteroperability* comes next: getting applications that previously had proprietary file formats to open up and talk to each other. This extends the life of both the applications and the data.

I*ntegration*: using this data coming from lots of different sources together to create new products that you never really imagined before.

I*ndependence*: I mentioned platform independence, but also device independence. People are using PalmPilots, browsers and ATMs for financial applications, and XML allows you to present this information to all these different devices.

JDJ: *What's IBM's role in helping to develop XML?*

Sutor: Two ways: first of all, in the W3C, which we think of as the core horizontal standards organization. We're involved with most of the key XML working groups such as XML Schema. I myself, for example, was one of the authors of the Document Object Model. On the industry side, traditionally we've been involved with a number of the industry-specific consortiums such as the OAG and the OMG, and recently such things as the Open Travel Alliance and the Financial Products Markup Language with one of our customers, J.P. Morgan. Really recently, one I'm personally very excited about – because it happened at OASIS – is XML.org. This is a brand new organization that has come together to act as an umbrella for developing standards.

Sagar: *Do you feel that at some point XML may replace EDI?*

Sutor: I expect that there will be a transition from using traditional EDI formats to more open XML standards that are being developed. In the meanwhile, there are a lot of people doing conversions between the old formats and the new ones. One example is the Financial Information Exchange. They've come up with very nice methods for preserving the old-style methods while moving to a new XML format.

Segar: *Some people are seeing it as coming in and solving all the data-formatting problems. And you know, being in the industry long enough, nothing really solves all the problems. There's a lot of hype associated with XML and then there's some reality. Would you elaborate on what you think may be the direction it's going to take and maybe what problems it's not going to solve?*

Sutor: It's an enabling technology, so it doesn't solve the problems by itself. It allows you to develop new markup languages. But you have to create those correctly and they should be done in a vendor-neutral way. For example, there's a fair amount of data modeling that should be done to produce a good XML DTD or schema. If it's done badly, you're going to have a bad product. A lot of what we're doing is introducing people first of all to the technology so they're aware of it, and then showing them how they can really use XML in the best possible way to integrate it with their products. From a developer's point of view, getting started with XML is very easy. If you go back to your EDI question, parsing out that EDI information and getting the information can be tricky because it's all encoded very tightly, although that might be efficient for moving the data around. XML is simple to read and it's easy to write new applications. There are parsers such as our XML4J that are available for free right now so the barrier for entry is very low. I think what's really generating a lot of the excitement about it is that there are so many tools available, such as those on our alphaWorks site. So the developers are very excited, but there's real work to be done by industry groups to create high-quality DTDs.

Sagar: *One of the things you mentioned earlier was that XML is now going out across the wireless, PalmPilots; there's WML; there's 3XML, WIDL. One of the dangers I see coming up is that there will be so many vertical definitions of XML in industry that it's going to be a hard thing to actually keep a tight rein on what is going on. Is OASIS planning to help solve such problems? Do they have some strategy as to how to control what becomes standardized?*

Sutor: Sure, the strategy is summed up by XML.org. We see XML.org as being in many ways complementary to W3C but for industry-specific standards. This means that XML.org can act as an umbrella for bringing together people to work on standards in industries. Thus there would be less need to set up all the ".org" XML Web sites we've been seeing. They could work within XML.org and have, for example, a human resources XML working area and write their standards there. It's actually a little trickier because there are many fine consortiums already out there that have developed standards. We are developing liaison relationships with many consortiums, but since things are being worked out, I don't want to talk about the details right now. If there's another organization that we can look at and say, "You have really done this in the right way. You have brought in enough people in the industry. People are agreeing to this and it has real, practical applications and implementations are happening," then in some sense we would like to liaison with them and be able to recommend their work.

JDJ: *Getting back to open standards. Why should developers care about open standards?*

Sutor: I've been a developer for a long time and I know how easy it is to create proprietary data formats. XML provides a really easy way of providing data in a format that can be changed easily and reused by other applications. Open standards are good because of the interoperability and integration aspects I mentioned before. In the long run standards lower the support costs because fewer technologies are involved. The data and the application will both be useful for a much longer time and the data will not be tied to particular platforms. This is important because 70% of the Fortune 1000 companies use three or more server platforms. Just as Java means portable programs, XML means portable data.

Sagar: *Since this interview is coming out in JDJ, I'd like your opinion on the relationship between XML and Java. Do they play well together? And what is the role that OASIS or XML.org will play in defining the standards for the Java industry?*

(continued on page 75)

Elixir

www.elixirtech.com

JDJSTORE.COM GUARANTEED LOWEST PRICES!

Take a look at our specials this month!

EASTLAND DATA SYTEMS

Internet Shopping with Java Shopping Cart

…Described as the most progressive and interactive form of shopping on the web today…This Java Applet provides a complete user interface package for Internet Shopping Web Sites. Using Java technology we produce a drag-and-drop shopping user interface that is fun and easy to use, encouraging shoppers instead of frustrating them with confusing controls that are hard to follow. And the easier it is to shop, the more you sell.

. . . . . $294.99

Hybrid Shopping Cart

This Java Applet provides a complete user interface package for Internet Shopping Web Sites. A "Hybrid" is defined as an offspring of two varieties. A blending of the best features from our CGI and Java shopping products, we took the most powerful aspects of Java technology; real-time, on-screen updating and computational capabilities. And combined those with the most desirable features of our CGI shopping Cart, namely it's flexibility and compatibility with web designers with artistic talent.

. . . . . $294.99

CGI Shopping Cart

The Shopping Cart automates the Shopping Process to make shopping on your site intuitive, straight forward, and enjoyable! It's one of the most affordable Shopping Carts because it was designed for small businesses. Specifically for entrepreneurs who are testing the Internet waters, and can't or don't want to make large investments into bells and whistles for their site. But simply want to make shopping on their site easy for the customer.

. . . . . $294.99

Guaranteed Best Prices

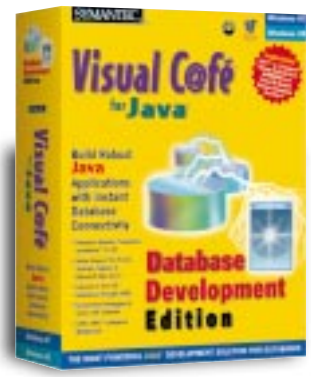
JDJ Store Guarantees the Best Prices.
If you see any of our products listed anywhere at a lower price, we'll match that price and still bring you the same quality service.

Terms of offer:

- Offer good through August 30, 1999
- Only applicable to pricing on current versions of software
- August issue prices only
- Offer does not apply towards errors in competitors' printed prices
- Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

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



SYMANTEC's

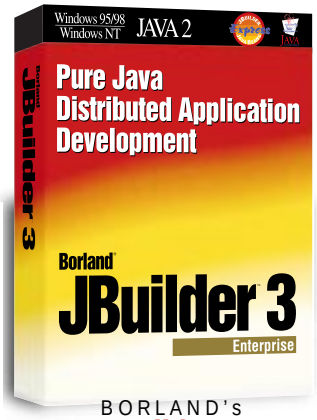
Visual Café Professional Edition

JDJStore.com Price . . $258.99
COMPARE...
Programmer's Paradise Price $259.95
Beyond.com Price $299.75

IBM's

VisualAge for Java 2.0 Enterprise Edition

JDJStore.com Price . . $2,498.99
COMPARE...
Programmer's Paradise Price $2,499.00
Beyond.com Price $2,525.00



BORLAND'S

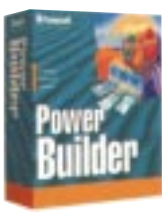
JBuilder 3 Professional New User

JDJStore.com Price . . $539.99
COMPARE...
Programmer's Paradise Price $545.99
Beyond.com Price $548.99

SYBASE

PowerBuilder Enterprise v.7

Release 7.0 of the market-leading enterprise development environment offers significant productivity enhancements and broad support for Web-based component standards. With the new HTML DataWindow, you can deploy to all major Web browsers. Tight integration in both development and deployment with EAServer offers highly competitive reliability, availability, and scalability (RAS) for PB applications developed for the Web.



PowerBuilder Enterprise v.7 . . . . . $2789.99

SYBASE

PowerJ Enterprise

PowerJ provides a true end-to-end solution for building sophisticated Internet applications, exploiting the benefits of HTML, Java clients, and delivering powerful Java server-side components. PowerJ not only offers powerful database capabilities -- it also integrates seamlessly with Sybase Enterprise Application Server, enabling enterprise-class applications from creation, to testing and debugging, to deployment.

PowerJ $304.99

SYBASE

SQL Anywhere Studio

JRun is the industry-leading tool for deploying server-side Java. JRun is an easy-to-use web server 'plugin' that allows you to deploy Java Servlets and JavaServer Pages. Servlets form the foundation for sophisticated server-side application development. Java servlets are platform independent, easy to develop, fast to deploy, and cost-effective to maintain.

SQL Anywhere Studio 1 user price $335.99
SQL Anywhere Studio 5 user price $849.99

SYBASE

Adaptive Server Enterprise for WIN NT

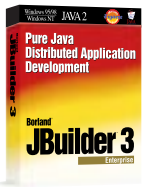
(Base product)

With the release of Adaptive Server Enterprise 11.9.2, Sybase introduces for - the first time - row-level locking (RLL) capabilities designed to provide faster performance, fewer deadlock contentions, and greater flexibility in the management of system resources. In addition to the new locking schemes, the Adaptive Server Enterprise 11.9.2 will include enhancements to the optimizer, query processing improvements such as index statistics and descending keys, database recovery enhancements and improved space management features.

Adaptive Server Enterprise $845.99

BORLAND

JBuilder 3 Professional Edition



JBuilder 3 Professional Edition is the most comprehensive set of award-winning visual development tools for creating platform-independent business applications, Pure Java, servlets, applets and JavaBeans for the Java 2 platform. Only JBuilder 3 combines the productivity of 300+ reusable JavaBeans with source, the ease of visual drag-and-drop Java 2 JFC/Swing development, an automatic database Application Generator and scalable database tools with JDBC connectivity.

Borland JBuilder 3 Professional Comp/Version Upgrade . . $264.99
Borland JBuilder 3 Professional New User $544.99
Borland JBuilder 3 Standard $84.99

KL GROUP

JProbe Suite

JProbe Profiler is the most powerful tool available for finding and eliminating performance bottlenecks in your Java code. JProbe Coverage makes it easy to locate individual lines of untested codes and reports exactly how much of your Java code has been tested. JProbe Threadalyzer lets you pinpoint the cause of stalls and and deadlocks in your Java applications and makes it easy to predict race conditions that can corrupt application data.

JProbe Profiler w/Standard Support (includes JProbe Memory Debugger) $464.99
JProbe Suite w/Standard Support $934.99

INSTANTIATIONS

JOVE Starter Kit

The JOVE(tm) Super Optimizing Deployment Environment(tm) lets you create and deploy the world's fastest, most efficient Java applications. JOVE combines aggressive whole-program and object-oriented optimization technologies, native compilation, and a scaleable runtime architecture and deployment environment. The runtime system includes state-of-the-art multi-threaded generational garbage collection, native multi-threading, low overhead polymorphism, and a number of other incredible technology pieces. As a result of all this technology, JOVE enables the creation of very high performance, robust executable files for the deployment of very large, complex Java applications.

JOVE Super Optimizing Deployment Environment. . . . $4494.99

SYMANTEC

VisualCafé Professional Edition

The award-winning, intuitive environment provides the utilities, tools and wizards for increased productivity like easy to use Interaction and JavaBeans Wizards and Editing. Support for the latest technology such as JavaBeans, servlets and JFC/Swing keeps advanced technology within your grasp. The integrated high performance compiler and debugger provides robust and fast tools for development while advanced features such as visual support for interaction and JavaBeans development and editing make complex development tasks extremely easy.

VisualCafé Professional Edition $258.99
VisualCafé Professional Edition (upgrade) $132.99

IBM

WebSphere Application Server

Whether it's business integration, Web self-service or e-commerce, IBM WebSphere Application Server provides the software and tools for building and deploying Web-based applications. With Standard, Advanced and Enterprise editions, Application Server supports all your e-business needs, from simple Web transaction processing to enterprise-wide Web applications. An open, extensible solution providing the highest levels of performance, security, availability and scalability, WebSphere Application Server leverages your existing IT investment to create new opportunities - now and into the future

WebSphere Application Server Standard Edition v2.02 . . . . . $668.99

INSTALLSHIELD

InstallShield Java Edtion 2.5

InstallShield Java Edition 2.5 is the powerful tool developers require to produce bulletproof InstallShield installations with Java versatility. You can target your application for multiple systems with cross-platform distribution. And InstallShield Java Edition 2.5 offers the key features and functionality designed to let developers go further in distribution and deployment.

InstallShield Java Edition $474.99

IBM

VisualAge for Java 2.0

With IBM's award-winning Java development environment, VisualAge for Java, you can build Web-enabled enterprise applications. It's a key element of the IBM e-business Application Framework. VisualAge leads the industry with proven support for building and testing Java applets, and Enterprise JavaBean components and servlets. It's the only Java development environment that supports the development and management of Java programs that can scale from Windows NT to OS/390 application servers.

VisualAge for Java, Enterprise. $2,498.99
VisualAge for Java, Enterprise Upgrade. $1,258.99
VisualAge for Java, Pro Edition $82.99
VisualAge for Java, Pro Edition Competitive Upgrade $25.99

ALIVE.COM

Alive e-Show 1.1

Alive e-Show is an easy, cost effective desktop application that allows non-technical PC users to create and publish streaming media e-shows to the Web. Using Alive e-Show, employees can create new content, integrate existing media, and publish it to the Web with a single click. It's a great way to integrate slides, digital photography, animation, synchronize audio and video, add hyperlinks, and supplement with closed captions. Create e-shows quickly and easily. You create slides in outline mode, record audio and video, and format the e-show using predefined templates with slide styles and placeholders for closed captions too. Allaire HomeSite® is included free, further simplifying any extra customization you may need using this HTML editor.

Alive e-Show 1.1 . . . . . $558.99

ORACLE

JDeveloper Suite Personal Edition

Oracle JDeveloper Suite 2.0 provides a complete Java development environment for developing and deploying applications ranging from Java and HTML clients to server-based business components for the Internet computing platform. JDeveloper 2.0 supports application development based on industry-standard Enterprise JavaBeans (EJB) and CORBA component models. Using intuitive and easy to use wizards, the product provides full support for a rich set of Java standards -- servlet, JDBC, SQLJ, InfoBus, and JFC/Swing. JDeveloper 2.0 comes with a servlet engine that allows servlets to be developed and tested within JDeveloper.

JDeveloper Suite Personal Edition. $119.99

GALILEO DEVELOPMENT SYSTEMS

Intr@Vision Foundation

Intr@Vision Foundation helps bring ColdFusion development to the next level. It provides an out-of-the-box application security architecture for handling your most complex intranet and extranet needs. Instead of spending 30% of your development time adding security to every application you build, it gives you a proven solution with a single line of code. Intr@Vision Foundation allows your developers to focus on building business solutions, not infrastructure.

Intr@Vision Foundation $3499.99

ALLAIRE

ColdFusion 4.0

JBuilder 3 Professional Edition is the most comprehensive set of award-winning visual development tools for creating platform-independent business applications, Pure Java, servlets, applets and JavaBeans for the Java 2 platform. Only JBuilder 3 combines the productivity of 300+ reusable JavaBeans with source, the ease of visual drag-and-drop Java 2 JFC/Swing development, an automatic database Application Generator and scalable database tools with JDBC connectivity.

ColdFusion 4.0 . . . . . $354.99
SkillBuilding with ColdFuion Interactive Training CD. . . . $284.99

ALLAIRE

HomeSite 4.0

HomeSite is the award-winning HTML editing tool that lets you build great Web sites in less time, while maintaining Pure HTML. Unlike WYSIWYG authoring tools, HomeSite gives you precise layout control, total design flexibility and full access to the latest Web technologies, such as DHTML, SMIL, Cascading Style Sheets and JavaScript. HomeSite 4.0 is the only HTML editor featuring a visual development environment that preserves code integrity.

HomeSite 4.0 . . . . . $87.99

Hungarian Notation with Java

A technique for attaining maintainable code

WRITTEN BY
BRIAN FARN

Creating software can be considered an art form, requiring all of the characteristics associated with an artist, such as creative style. Most artists, however, aren't required to modify their creations after the work has been purchased. Software, on the other hand, needs to be maintained either by its creator or by others if the creator has moved to other responsibilities.

The maintainability of software is an increasingly important issue. As industry demand for programmers' skills increases, so does their turnover rate within a company, thus diminishing the likelihood of the original programmer's being available when a problem arises or when the product needs to be enhanced. Also, a programmer who produces volumes of code may need some time to refamiliarize him- or herself with code that was written only a month earlier, perhaps, but hundreds of lines ago.

Programmers exposed to code that has a style different from their own may institute parts of the new style into their own code if there's an obvious advantage. Most programmers restyle inherited code to their own conventions. New programmers should thus be exposed to good coding style and its advantages before any code is written.

Hungarian Notation

One useful coding technique benefiting both code readability and writability – and hence maintainability – is Hungarian Notation. This technique is suitable for C/C++ programs that make use of pointers. It's also applicable to Java programs. Essentially, Hungarian Notation is a naming convention that allows a programmer to determine the type of variable or constant just by looking at its name. The letter "i", for example, can be used to denote an "int". By prefixing an "int" variable named "count" with an "i" as in "iCount", a programmer can tell by looking at the variable name that the variable is of type "int".

As an example, look at the following code segment:

```
float x = 0;
int   y = 0;

x = y;
```

Looking only at the statement "*x* = *y*", someone attempting to understand this code might assume that both variables are of the same type. A good guess would be that *x* and *y* are coordinate variables of type "int". However, we don't know if *x* and *y* are objects, numbers or characters. This is especially true if the declarations for *x* and *y* are not physically close to the assignment statement. You'd need to search through the code looking for the declarations. If the original programmer had used Hungarian Notation, the statements would look like this:

```
float fX = 0;
int iY = 0;

fX = iY;
```

A person reading just the statement "fX = iY" can deduce that an "int" is being assigned to a "float" and may suspect that the code is a little out of the ordinary. Now look at these statements:

```
int   iX = 0;
float fY = 0;

iX = fY;
```

From reading just the statement "iX = fY", a person can now deduce that a float is being assigned to an integer and be very concerned. Because of a type mismatch this program won't compile successfully.

As a side benefit, Hungarian Notation allows the original programmer to recognize that an explicit cast is required in an assignment statement while the code is being written. A programmer writing the above statement would notice that "iX" and "fY" were of different types and would insert a cast "(int)" before entering the right side of the assignment, as in "iX = (int)fY", or might change "iX" to a float variable, "fX".

This is the writability aspect of using Hungarian Notation, in which errors are detected and removed as the code is written. Discovering errors earlier in the development cycle is a natural benefit.

Reuse of Variable Names

Hungarian Notation also allows variable names to be reused when only the notation part of the name is different. Let's take a look at a slightly more complex example, shown in Listing 1. In this class the constructor is given a file name and a "boolean." The file name is the name of a file and the boolean is used to tell the constructor whether the file name should be used.

In this example an underscore character prefix distinguishes the instance variables from the local variables and function arguments. Also, the letter "b" is used as the notation prefix for boolean type variables, as in "_bNameIsValid". The variable name "strFilename" is used for both the input argument and the instance variable with the exception of the underscore prefix in the instance variable. If you just see the statements:

```
_strFilename  = strFilename;
_bNameIsValid = true;
```

you can tell immediately which variables are instance variables, booleans or strings, and – if the original programmer used sensible variable names – the purpose of each variable. Most important, the assignment statements look correct from a "type" point of view: a "String" is assigned a "String," and a "boolean" is assigned a "boolean" constant.

The same variable name can also be used for related variables. The Hungarian Notation prefix provides a distinction between the data type of the variables. For instance, the following two state variables represent the string for a "Cancel" button and the Button object itself.

```
   String strCancel = new String();
// The 'Cancel' string
   Button btnCancel = new Button(
strCancel );  // The cancel button
```

The following statements ask a user to enter a user ID. A JLabel is used to identify a JTextField as the place to enter the user ID. Both variables are named "Userid" but are distinguished by their particular data type notation.

```
   JLabel labelUserid = new JLabel
( "Enter your user id here:" );
   JTextField fieldUserid = new
JTextField();
```

Search and Replace

When writing new code or upgrading existing code, notated variables are readily modifiable. For instance, a text editor's replace utility can rename all instances of a variable named "iW" to "iX" in one operation. However, the same operation can't be used to rename variable "w" to "x" without errors as all instances of the letter "w" would be replaced whether or not it represents a variable.

Variable Naming Conventions

The following is a suggested set of rules for generating variable names and is not meant to be a rigid set.

1. The prefix should be determined from the datatype definition.
2. The prefix should be in lowercase only.
3. If the variable is declared inside a class as instance data, the prefix should be prefixed by an underscore as in "_fVarName".
4. All words in the variable name should be capitalized to increase readability, starting with the first letter after the notation prefix.
5. Don't overdo it. If you have a single JPanel, name it "panel."
6. For longer type names such as FontMetrics, use either an acronym such as "fm" or a portion of the name such as "metric."

Here are some name prefix suggestions:

Primitive Types

TYPE	NOTATION	EXAMPLE
int	i	iIndex
short	s	sIndex
char	c	cLetter
boolean	b	bFound
byte	y	yData
float	f	fInches

Arrays

Append the letter "a" to the type prefix:

```
byte[]   yaNumbers;
int[]    iaNumbers;
String[] straNames;
```

Object Types

```
Integer          intIndex;
String           strIndex;
StringBuffer     strbIndex;
Point            ptIndex;
Boolean          boolFound;
Label            labelTextString;
TextField        fieldName;
Panel            panelTextArea;
Frame            frameWindow;
Window           wndPrompt;
Popup            popSelection;
MyOwnObjectType  mootPoint;
```

Container Objects

Prefix a notation for the object type to the notation for the variable type.

```
   Vector vintIndices;
// Vector of Integer objects
   Hashtable hashserAddresses;
// Hashtable of Serializable address
   objects
```

Instance Variables

Add an underscore before the type prefix and the variable name.

```
byte[]  _yaNumbers;
int     _iIndex;
boolean _bFound;
float   _fInches;
String  _strName;
```

As an alternative, another designation such as "m_" can be added to the beginning of a variable name to indicate an instance (member) variable.

```
byte[]  m_yaNumbers;
int     m_iIndex;
boolean m_bFound;
float   m_fInches;
String  m_strName;
```

Class Variables

```
Prefix a "c_" to the name.

int     c_iIndex;
double  c_dValue;
```

A Life Example

Say tomorrow someone came to you and said a fellow developer is unable to complete a critical piece of code and you're the only person who has the skills and knowledge to finish the job – and by the way, you only have until the end of the week to complete the task. Otherwise the customer will cancel the contract and take that six-figure sum elsewhere.

Before you panic, you decide to evaluate the situation by reading a sample of the code. You need to find out whether it looks reliable and whether you can understand it well enough to modify it. The answers to these questions, multiplied by a hundred, will give you a good idea of how you'll react when asked to inherit the entire class library. You arbitrarily turn to the function shown in Listing 2. What does it do?

This code is actually part of a text editor class that searches for a given string with optional case sensitivity within a vector of string buffers beginning at an *x* and *y* coordinate. The "result" returned is the point coordinate of the found text, *y* being the row and *x* being the column, or (-1, -1) if the text string isn't found. From this code we can deduce several things:

1. The original programmer was kind enough to use understandable variables, not just "s"or "zx".
2. The variables "numberOfLines," "lines" and "cursor" aren't declared in the function so they're probably instance or class variables, but we're not really sure and we'll have to hunt for them.
3. The author may have had difficulty with variable naming since there are similarly named variables "line," "lines" and "line1".
4. The variable "lines" is probably a "Vector" since on line 25 we see a "lines.elementAt()" function call. Then again, it could be a "DefaultListModel".
5. The variable "cursor" is probably a "Point" since on lines 8 and 9 we see "cursor.x" and "cursor.y". However, it could also be an "Event" object or, less likely, a "Rectangle" object.
6. The variable "numberOfLines" is probably an "int" since it's being assigned to another "int" on line 23.

Looking at the individual statements from line 25 to line 29, it's difficult to determine whether the type assignments are correct and the method calls against the variables are appropriate. Could this be the real source of the term *fuzzy logic*?

Now compare the code in Listing 2 to the same code in Listing 3, which has been written using Hungarian Notation.

Several things are worth noticing here:

1. The "cases" variable has been replaced with "bCase", which clearly defines the variable as a "boolean" intended to mean case sensitivity. Since "case" is a keyword, the original author couldn't use it as a variable name, which is probably why the variable was originally named "cases".
2. The variables "_iNumberOfLines", "_vstrbLines" and "_ptCursor" are clearly defined as instance variables of type "int", "Vector" and "Point", respectively.

3. The variables named "line", "lines" and "line1" have been replaced by "strbLine", "vstrbLines" and "strLine", signifying a "StringBuffer", a "Vector" of StringBuffers and a "String". The same root name "Lines" has been reused for all three variables, which eliminates the extra step of having to think up different but similar variable names that represent the storage of a text string.

Each statement from line 25 to line 29 can now be examined individually, out of context, to determine whether each line's syntax is correct. For example, let's visit line 25.

```
25 strbLine = (StringBuffer)_vstr-
bLines.elementAt( i );
```

Here's what we can deduce:

1. The function "elementAt()" is correctly called against the object "_vstrbLines" since the "v" indicates a "Vector".
2. The argument type to the function "elementAt()" is correct since the variable "i" is an "int".
3. The "(StringBuffer)" cast against "_vstrbLines.elementAt(i)" is correct since the "Vector" holds objects of type "StringBuffer" as indicated by the "strb" in "_vstrbLines".
4. The left side of the assignment statement is also correct since the result of the cast is being assigned to a "StringBuffer" as indicated by the "strb" in "strbLine".

The statement is clear and easier to read because it contains more information than the original. More important, it's easier to write because as a developer who uses Hungarian Notation consistently, you can tell the statement is correct as you write it.

Summary

Differences in artistic coding style can vary greatly among programmers. The personal aspects of coding style are associated with pride of ownership. However, a lack of standards can indirectly produce code that has unknown reliability characteristics and is hard to maintain. The result can be unplanned work of unknown duration, which includes tasks ranging from documenting to rewriting. It's financially beneficial to use accepted coding techniques, such as Hungarian Notation, consistently. This will provide long-term time savings. Its use can result in code that is more readable, writable, understandable and thus more maintainable, transferable and reliable. Think about this the next time you inherit someone else's code or would like to pass on some of your own code to that person who's looking over your shoulder.

farn@ca.ibm.com

BENEFITS OF USING HUNGARIAN NOTATION

Readability:	Notation indicates variable types, and scope. Assignment statements can be verified to be correct without the need to hunt for variable declarations. Variables can be determined to be instance variables by looking at the variable name.
Writability:	Coding errors are discovered sooner. Programmer understands that the variable type and scope is correct while writing it. Required casting is easily identified. Variable name roots can be reused.
Maintainability:	Code is understood more quickly by both the original programmer, and future programmers.

AUTHOR BIO

Brian Farn researches and develops Java software at IBM and is currently associated with the IBM VisualAge for Java team at the IBM Software Solutions Toronto Laboratory. He is a graduate of the University of Western Ontario with degrees in physics and electrical engineering.

Listing 1

```
public class Sample
{
  String  _strFilename  = null;
  boolean _bNameIsValid = false;

  public Sample( String strFilename,
boolean bUseFilename )
  {
    // Store the filename if it is
valid, and if told to
    //---------------------------------
---------------
    if( bUseFilename == true && str-
Filename != null )
    {
      if( strFilename.length() > 0 )
      {
        _strFilename  = strFilename;
        _bNameIsValid = true;
      }
    }
  }
}
```

Listing 2

```
1  public Point find( String find,
   boolean cases )
2  {
3    Point result = new Point( -1,-1 );
4    StringBuffer line   = null;
5    String       needle = null;
6    String       hay    = null;
7    int          index  = 0;
8    int          x      = cursor.x;
9    int          y      = cursor.y;
10
11    needle = cases == false ?
12      find.toUpperCase() : find;
13
14     line = (StringBuffer)lines.ele-
       mentAt( y );
15
16     if( x >= line.length() - 1 )
17     {
18      x = 0;
19      ++y;
20      if( y >= numberOfLines ) return
        result;
21     }
22
23     for( int i=y; i<numberOfLines;
     ++i )
24    {
25     line = (StringBuffer)lines.ele
       mentAt( i );
26     String line1 = line.toString();
27     hay = cases == false ?
28     line1.toUpperCase() : line1;
29     index = hay.indexOf( needle, x );
30
31     if( index != -1 )
32     {
33       result.x = index;
34       result.y = i;
35       break;
36      }
37
38      else x = 0;
39    }
40
41    return result;
42  }
```

Listing 3

```
1 public Point find( String strFind,
  boolean bCase )
2 {
3  Point  ptResult = new Point( -1,-1 );
4  StringBuffer strbLine  = null;
5  String       strNeedle = null;
6  String       strHay    = null;
7  int          iIndex    = 0;
8  int          iX        = _ptCursor.x;
9  int          iY        = _ptCursor.y;
10
11 strNeedle = bCase == false ?
12  strFind.toUpperCase() : strFind;
13
14 strbLine = (StringBuffer)_vstr
   bLines.elementAt( iY );
15
16 if( iX >= strbLine.length() - 1 )
17 {
18   iX = 0;
19   ++iY;
20   if( iY >= _iNumberOfLines ) return
     ptResult;
21 }
22
23 for( int i=iY; i<_iNumberOfLines; ++i )
24 {
25  strbLine = (StringBuffer)_vstr
    bLines.elementAt( i );
26 String strLine = strbLine.toString();
27 strHay = bCase == false ?
28  strLine.toUpperCase() : strLine;
29 iIndex = strHay.indexOf( strNeedle, iX );
30
31   if( iIndex != -1 )
32   {
33    ptResult.x = iIndex;
34    ptResult.y = i;
35    break;
36   }
37
38   else iX = 0;
39  }
40
41 return ptResult;
42 }
```

CODE LISTING

Code listings for this article can also be located at www.JavaDevelopersJournal.com

Cerebellum 1.3

by Cerebellum Software

An innovative middleware and development tool for accessing disparate databases

WRITTEN BY JIM MATHIS

AUTHOR BIO

Jim Mathis is a freelance Java and JavaScript consultant by night and a communications system architect by day. He has been active in the Internet community from the very beginning and wrote one of the first implementations of TCP/IP. A former Apple employee, Jim concentrates on Macintosh as a platform.

jmathis@ais.net

Cerebellum 1.3: Cerebellum Software Inc.
Web: www.cerebellumsoft.com
Phone: 1-888 862-9898

Requirements:
Java 2 (Windows 95/98, Window NT, Solaris for SPARC and x86), 32 MB RAM, 10—30 MB disk

Pricing:
$995 for a single developer's license; runtime license varies depending on the number of servers and concurrent users. Enterprise packages start at $40,000.

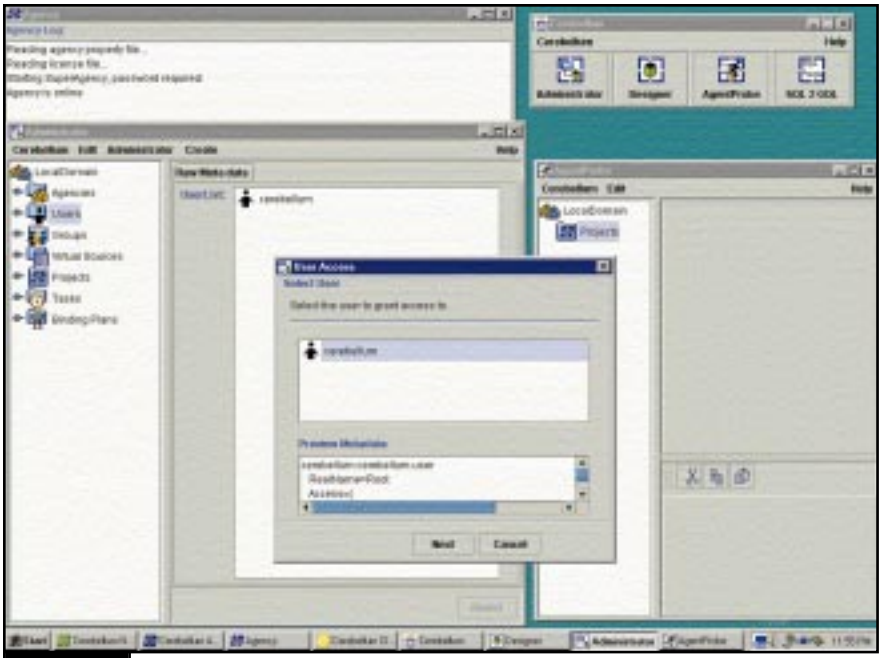


FIGURE 1: Administrator tool

Cerebellum Software Inc. recently announced the latest release of their innovative data access middleware and query application development environment – Cerebellum 1.3, which replaces the tedium of coding SQL queries with a drag-and-drop visual programming approach. Using Cerebellum, queries are saved as *agents*, which are units of compiled Java code that communicate with data sources in their native tongue. These agents are managed by servers called *agencies* that mediate between the workstations that run the query–client applications. This middleware approach provides a unified way to access, integrate and manage data from a variety of databases and data sources at different physical locations. Cerebellum is based on Java 2 Enterprise Edition technology and incorporates features from CORBA.

Installation

I tested Cerebellum in a single workstation environment on a mid-range Pentium II-266 workstation with 128 MB RAM and running Windows NT 4.0 SP3. Full installation of Cerebellum with documentation and examples required 30 MB of disk, not counting the necessary JDK 1.2.2 and database packages. (I used MS Access and the shareware version of MySql.) Cerebellum runs on any platform supporting Java 2 with installation instructions provided for Windows and UNIX systems.

Cerebellum is designed from the start to run in a complex business, corporate or enterprise network environment. The major components – NameServer, Agency and Client – can run on the same or different machines. The terminology used by Cerebellum is a little confusing initially. The NameServer (different from a Domain Name Server) provides the central repository for information linking the objects in a Cerebellum system. An agency runs the agents that translate to the actual data sources: either databases or external applications.

The standalone installation script simplifies configuring Cerebellum for use on a single machine, which is typical of a smaller business setting. The advanced configuration provides flexibility in placing the NameServer and Agencies on different workstations to support large enterprise solutions. I installed my standalone single workstation configuration in about 10 minutes. Complex configurations involving multiple databases and servers can take several hours to plan and install. I recommend that a new user first experiment with the standalone environment and prototype a few queries before proceeding to plan and install a full enterprise configuration.

The value of a data access middleware product is its ability to work with many data sources. Cerebellum supports relational database products from Oracle, Sybase, Informix and Microsoft, and the shareware MySql database. A JDBC–ODBC bridge connects Cerebellum to any database that has an ODBC driver.

Cerebellum supports interaction with applications running on IBM's mainframe CICS and imports information stored as ordinary files. Support is planned in future versions for access to data from SAP applications, object database management systems, Enterprise JavaBeans and LDAP directories. Because of Cerebellum's data-independent architecture, existing Cerebellum queries can be modified easily to use these new data sources.

Creating a Query

Before we get to build our queries, we need to perform some administrative functions

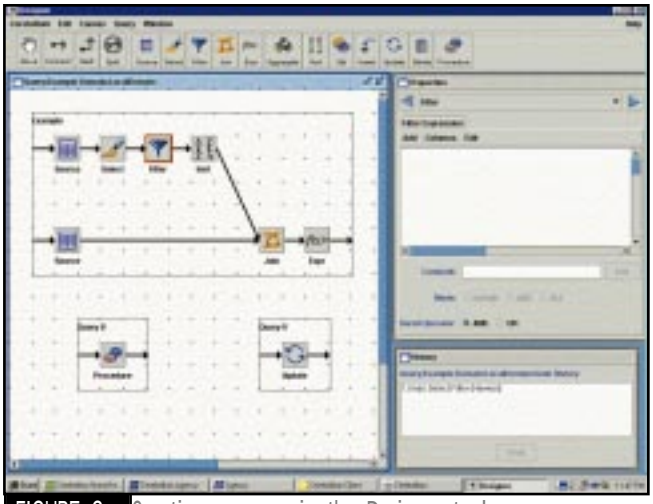


FIGURE 2: Creating a query in the Designer tool

using the Administrator tool (see Figure 1). Cerebellum's integrated data access approach is based on a common virtual data model. A binding plan specifies how columns in the virtual data model (the virtual source) map to specific columns in specific databases (physical sources). This binding plan deals with the fact that the same data columns may be called differently in different databases. We must identify data that's semantically the same even though labeled differently (e.g., "Full Name" and "Employee Name"), and different even though labeled the same (e.g., "Name" of product and "Name" of a contractor). Once the binding plan has been created, it's reused in future queries. The Administrator tool also provides functions to organize users, groups and projects, and to assign access privileges.

Once the setup functions are complete, you're ready to create queries using the Designer user interface tool. Query applications are developed, without writing any SQL code, by dragging and dropping a combination of 12 "glyphs," the basic units of Cerebellum's graphical query language, onto a worksheet or canvas. Each glyph represents a predesigned code fragment that performs a specific action (see Figure 2). You connect glyphs together with arrows to represent the flow of data through a query. A query can be as simple as a single glyph that retrieves data from a particular source, or it can involve multiple sources, joins, manipulations and transactions. Different glyphs do different sorts of work. Some serve as sources of data; others extract data; still others manipulate data, delete data from data sources or write data to data sources. The 12 glyphs are source, select, filter, join, expression, aggregate, sort, set operation, insert, update, delete and stored procedure. While Cerebellum hides the differences in SQL dialects across different database vendors, some basic knowledge of SQL and database query techniques is definitely useful to make the most effective use of Cerebellum.

When complete, the query is stored, ready to use as a standalone query application, or it can generate an agent to be called from a Web page or used as a component of another application. Cerebellum 1.3 features three APIs for developers: C++/CORBA, JDBC and ActiveX. Cerebellum provides a set of Java libraries that allows Java programs access to query creation and execution features.

Conclusion

I've touched on just a few of the very powerful features provided by Cerebellum. One limitation is the lack of integration with configuration and version control systems, often standard in enterprise settings. Cerebellum is most effective in situations that require integration of data from more than two databases. The greater the number of different databases or data servers, the more Cerebellum will simplify your work and improve your productivity.

XML DTD for EJB Deployment Descriptors

Sun moves toward a goal of interoperability for enterprise beans among EJB vendor products

WRITTEN BY
JASON WESTRA

To those of you familiar with Enterprise JavaBeans (EJB), deployment descriptors are nothing new. Essentially, a deployment descriptor's purpose is to collect declarative information that can be modified during deployment of an enterprise bean. Deployment descriptors are a key element in the component-based development capabilities of EJB. They allow users to modify, link and deploy EJB in a graphical environment rather than having to perform low-level code changes to reuse a component. The latest public draft of EJB specification 1.1 includes sections on the XML DTD for deployment descriptors, an important step toward enterprise bean portability between EJB servers.

In the spirit of ***JDJ***'s focus on XML this month, I'll cover some important pieces of the newly released XML DTD for deployment descriptors in EJB specification 1.1 (section 16.6). As this is being written I have yet to find a production EJB server that supports the DTD (if you know of one, please let me know!); however, I'll include a "best shot" example descriptor using the DTD so you get a feel for the format of its metadata. The example is based on the Intergalactic Ticket System's TicketEntityBean from September's **EJB Home** column (***JDJ*** Vol. 4, issue 8). For reference see the online source code at the ***JDJ*** Web site.

A Look Inside the EJB's XML DTD

A deployment descriptor contains two kinds of metadata (data about data) for an enterprise bean:

- *Structural information*
- *Application assembly information*

Structural information describes specific characteristics of an enterprise bean such as whether it is an entity or session bean, and any external dependencies it might have. This information must be included in an ejb-jar file by its producer. Structural information about an enterprise bean is entered by the bean provider, and it's advisable not to modify this information during ejb-jar assembly or deployment. Table 1 lists common structural information about an enterprise bean that you might include in a deployment descriptor.

Application assembly information is the responsibility of the application assembler. It describes how an enterprise bean in an ejb-jar file is assembled into a coarse-grained deployment unit. This information isn't mandatory, and it may be modified at deployment time. However, be aware that an enterprise bean's behavior may be affected by these changes. Table 2 lists common application assembly information you might encounter in a deployment descriptor.

XML Element	Description/Value(s)
ejb-name	Logical name for enterprise bean
ejb-class	Fully qualified class name of the bean
home	Fully qualified class name of the bean's home interface
remote	Fully qualified class name of the bean's remote interface
entity, or session	Element for entity or session bean
session-type	Stateless or stateful
transaction-type	Container or bean
trans-attribute	How container manages transaction (i.e., Required, Mandatory, RequiresNew, etc.)
persistence-type	Container or bean, for entity bean only
prim-key-class	Fully qualified class name of primary key. Required only for bean-managed entities
cmp-field	Container-managed field, required if container-managed entity
env-entry	Environment entries, optional

TABLE 1: Common structural information

The EJB specification describes the responsibilities of roles during the life cycle of an enterprise bean's development and deployment. The roles of bean provider and application assembler are separate, and they are assigned different responsibilities regarding the creation/modification of a deployment descriptor. There will be overlap in the responsibilities of each role, however, resulting in an application assembler's possibly modifying structural information about an enterprise bean. I played each role in creating the example descriptor.

TicketEntityBean XML Deployment Descriptor

The EJB XML DTD (enough with the acronyms!) defines how to describe both session and entity beans. Some element tags, like env-entry – a description of a bean environment property – pertain to both bean types. Others are specifically for session or entity beans only. The TicketEntityBean XML descriptor example clarifies common entity bean elements for you. Listing 1 contains the complete source code of this XML descriptor.

As indicated earlier, the DTD contains structural information about an enterprise bean and may contain assembly information as well. I'll begin analyzing the structural information about the TicketEntityBean deployment descriptor and finish with the assembly information.

First, notice the top line of the ejb-jar descriptor in the TicketEntityBean descriptor. A well-formed, valid deployment descriptor must refer to the DTD with the statement:

```
<!DOCTYPE ejb-jar PUBLIC "-//Sun
<Microsystems Inc.//DTD Enterprise
JavaBeans 1.1//EN">
```

XML Element	Description
assembly-descriptor	Root element for application assembly information
security-role	Role name value for EJBObject.isCallerInRole(String roleName)
method-permission	Defines and binds permissions to access methods to security roles
container-transaction	Used to assign container-transaction elements to container-managed beans

TABLE 2: Common application assembly information

Sun Microsystems has plans to deliver an ejb-jar file verifier that will check for malformed XML as described in EJB specification 1.1, section 16.6. A verifier will provide bean providers and application assemblers the ability to validate their work, ensuring correct DTD semantics are upheld.

Next, I continue the XML descriptor with the ejb-jar element.

```
<ejb-jar>…</ejb-jar>
```

This element is the root of an EJB deployment descriptor, which may contain multiple enterprise beans. This month's example contains only one enterprise bean. The ejb-jar element may contain optional descriptions of the ejb-jar, its icon files, display name and an assembly-descriptor section, but it *must* include at least one enterprise bean element.

The enterprise bean element contains one or more declared enterprise beans: session or entity. The XML descriptor will not be valid if this element is empty.

```
<enterprise-beans>…</enterprise-
beans>
```

Within this section I enter the structural information about my TicketEntityBean.

TicketEntityBean's Structural Information

The entity element in my example contains the structural information that is the bean provider's responsibility to enter.

```
<entity>…</entity>
```

The entity element has mandatory and optional entries to allow the bean provider to add detail to the bean as needed.

Mandatory Entity Elements

Mandatory elements are the ejb-name, home, remote, ejb-class, persistence-type, prim-key-class and reentrant fields.

For all but the ejb-name element, I have simply copied the values from my previous deployment descriptor file from the September column into these elements as necessary. Thus you can see that the example's home element contains the fully qualified name of the entity bean's home interface class. Similarly, the remote and ejb-class hold the fully qualified class names of the remote and entity bean, respectively.

```
<ejb-name>TicketEntityBean</ejb-name>
<home>jdj.ticketing.containerman-
aged.TicketEntityHome</home>
<remote>jdj.ticketing.containerman-
aged.TicketEntity</remote>
<ejb-class> jdj.ticketing.container-
managed.TicketEntityBean </ejb-class>
```

Note: ejb-name is simply a logical name for the enterprise bean. It is not the JNDI that will be assigned by the deployer at a later time. Also, ejb-name must be unique for a given ejb-jar file.

Because TicketEntityBean is container-managed, my descriptor's persistence-type is "Container," and like the home and remote elements, I have entered the prim-key-class in fully qualified form.

```
<persistence-type>Container</persis-
tence-type>
<prim-key-class>jdj.ticketing.con-
tainermanaged.TicketEntityPK</prim-
key-class>
```

The last mandatory entry in the entity element is whether or not the bean is reentrant. I have listed my TicketEntityBean as False.

```
<reentrant>False</reentrant>
```

Optional Entity Elements

The entity element can optionally include a description, display name and icon files. Additional optional fields may be contingent on factors such as whether the bean is container-managed or bean-managed. These include cmp-field (container-manager field), prim-key-field, env-entry (a bean environment property), ejb-ref (references to other ejbs in the ejb-jar file), security-role-ref and resource-ref (reference to external resources). Because the TicketEntityBean is container-managed, I've included some of these elements in my descriptor.

This is an example of an optional environment property where the entity bean has declared an idleTimeoutSeconds environment property with a value of 5.

```
<env_entry>
 <env_entry-name>idleTimeoutSec-
onds</env_entry-name>
 <env_entry-type>String</env_entry-
type>
 <env_entry-value>5</env_entry-value>
</env_entry>
more env-entries possible…
```

Following is an example of a few of the container-managed fields declared for the TicketEntityBean:

```
<cmp-field><field-
name>arrivalCity</field-name></cmp-
field>
<cmp-field><field-
name>departDt</field-name></cmp-
field>
more cmp-fields possible…
```

Application Assembly Information

Table 2 describes numerous elements that an application assembler can include in the descriptor. All are optional; thus the assembly descriptor may be left out of the ejb-jar file.

The assembly-descriptor element –

```
<assembly-descriptor>…</assembly-
descriptor>
```

– can optionally contain information about security roles, method permissions and an enterprise bean's transaction semantics with its container. I have opted to have no role-based security on my TicketEntityBean, nor any special method permissions based on role. However, another ticket agency reusing this entity bean may decide to place a security restriction on the bean, allowing only TicketAgent roles to access it. In this case an entry would be made by the application assembler to include the security role element:

```
<security-role>…</security-role>
```

Likewise, they would want to include which methods are restricted to this new security role in the form of the following tag:

```
<method-permissions>…</method-permis-
sions>
```

I don't have security information to worry about; however, I do want to describe how my entity bean behaves within the context of a transaction. To do so I have to include an element, container-transaction. The full assembly-descriptor element is listed below.

```
<assembly-descriptor>
 <container-transaction>
  <method>
   <ejb-name>TicketEntityBean</ejb-
name>
   <method-name>*</method-name>
  </method>
  <trans-attribute>Required</trans-
attribute>
</container-transaction>
</assembly-descriptor>
```

In this example I list the entity bean in question (i.e., ejb-name), which methods are managed by the container (i.e., method-name) and exactly how they are managed (i.e., trans-attribute). For my purposes I am dealing with the TicketEntityBean, in which all methods signified by an asterisk (*) are required (Required) to operate within a transaction context.

Author Bio

Jason Westra is a managing partner with Verge Technologies Group, Inc., a Java consulting firm specializing in Enterprise JavaBeans solutions.

> "A standard allowing EJB developers and deployers to speak a common language will only increase the efficiency of EJB application development"

The assembly-descriptor adds character to your enterprise beans that is not or cannot be determined by the bean provider. As security or transaction context needs evolve, the assembly-descriptor enhances EJB's component-model capabilities by allowing modification of bean properties without code changes.

Summary

With the addition of a DTD for deployment descriptors in the EJB specification 1.1, Sun Microsystems is stepping in the right direction toward an end goal of interoperability for enterprise beans among EJB vendors' products. A standard allowing EJB developers and deployers to speak a common language will only increase the efficiency of EJB application development.

As with the first draft of any spec, there are holes in the DTD that still need to be filled. For example, there is no description of how container-managed fields map into their persistent storage, and at the time of this writing sections 16.4 and 16.5, deployer's responsibilities and container provider's responsibilities, haven't been specified. When these sections are detailed, an element to describe a bean's JNDI name, for instance, will become self-evident (no pun intended!). In the next column I'll cover other portability issues beyond XML, with regards to the EJB specification.

jwestra@uswestmail.net

Listing 1: TicketEntityBean XML descriptor

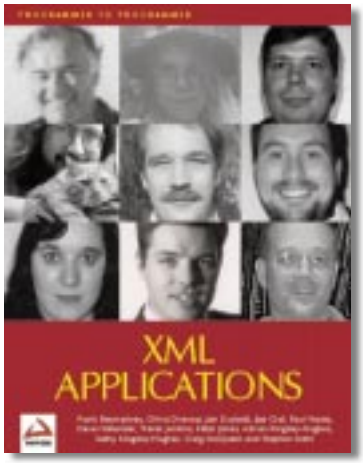
```
<!DOCTYPE ejb-jar PUBLIC "-//Sun <Microsystems Inc.//DTD
Enterprise JavaBeans 1.1//EN">
<ejb-jar>
<description>
This ejb-jar file contains the assembled enterprise bean(s)
that make up the
 Intergalactic Ticket System.
</description>

<enterprise-beans>
 <entity>
  <description>
The TicketEntityBean represents a ticket entry made by a
ticket agent. It is container managed and currently assembled
to deploy against a JDBC backend.
</description>

<ejb-name>TicketEntityBean</ejb-name>
<home>jdj.ticketing.containermanaged.TicketEntityHome</home>
<remote>jdj.ticketing.containermanaged.TicketEntity</remote>
<ejb-class> jdj.ticketing.containermanaged.TicketEntityBean
</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>jdj.ticketing.containermanaged.TicketEnti-
tyPK</prim-key-class>
<reentrant>False</reentrant>

<env_entry>
 <env_entry-name>maxBeansInFreePool</env_entry-name>
 <env_entry-type>String</env_entry-type>
 <env_entry-value>20</env_entry-value>
</env_entry>

<env_entry>
<env_entry-name>maxBeansInCache</env_entry-name>
 <env_entry-type>String</env_entry-type>
 <env_entry-value>1000</env_entry-value>
</env_entry>

<env_entry>
 <env_entry-name>idleTimeoutSeconds</env_entry-name>
 <env_entry-type>String</env_entry-type>
 <env_entry-value>5</env_entry-value>
</env_entry>

<env_entry>
 <env_entry-name>isModifiedMethodName</env_entry-name>
 <env_entry-type>String</env_entry-type>
 <env_entry-value>isModified</env_entry-value>
</env_entry>

<cmp-field><field-name>arrivalCity</field-name></cmp-field>
<cmp-field><field-name>departDt</field-name></cmp-field>
<cmp-field><field-name>price</field-name></cmp-field>
<cmp-field><field-name>flightNumber</field-name></cmp-field>
<cmp-field><field-name>arrivalDt</field-name></cmp-field>
<cmp-field><field-name>ticketNum</field-name></cmp-field>
<cmp-field><field-name>seatNumber</field-name></cmp-field>
<cmp-field><field-name>passengerNumber</field-name></cmp-
field>
<cmp-field><field-name>departCity</field-name></cmp-field>
 </entity>

<assembly-descriptor>
 <container-transaction>
  <method>
   <ejb-name>TicketEntityBean</ejb-name>
   <method-name>*</method-name>
  </method>
  <trans-attribute>Required</trans-attribute>
</container-transaction>
</assembly-descriptor>
</ejb-jar>
```

CODE LISTING

Code listings for this article can also be located at www.JavaDevelopersJournal.com

XML Applications

by F. Boumphrey, O. Direnzo, J. Duckett et al.
List Price: $49.99
649 pp
Wrox Press Ltd.; ISBN: 1-861001-52-5



This past March I got involved in writing a BizTalk interface to one of our business engines using XML. At that time I was familiar with the tagging scheme, the concept of DTDs and the different parser implementations that supported DOM and SAX. However, I hadn't had the privilege of writing an XML application. Like most software engineers who want to research a technology, I turned to the Web. As expected, I found tons of information about XML. So much so that it was overwhelming (DOM, SAX, XSL, XQL, DTD, Parsers, XML/EDI, CSS, etc.). Like any good employee trying to meet a deadline, I started to go through the various sites and absorb as much information as I could. Once I had achieved information overload, a buddy of mine who enjoys buying books purchased *XML Applications*. As much as I hate to reveal programming secrets, this book helped me understand the DOM hierarchy and its interfaces.

Chapter 1 started by introducing the value of XML followed by a description of what a well-formatted document looks like. Chapter 2 did a good job explaining XML declarations, elements and attributes and how they're used. The book then went into a comprehensive explanation about DTDs (document type declarations).

For me, the chapter that really made the book indispensable was 6, which explained in detail the XML Document Object Model and the DOM interface, a common API that allows the programmer to access the XML document as a tree model. I found the illustrations in this chapter to be invaluable. My only complaint about this chapter is that the example code was written in JavaScript and IE5 XML ActiveX control. I would have preferred Java examples that used the IBM or Sun Java parser.

As I continued reading, I noticed that the book didn't discriminate against languages. Chapter 11 provided a case study in which XML is used as a common data schema between vendors' Web sites. The case study was implemented leveraging VBScript and C++.

I finally found the content I was looking for in Chapter 12, the Java and XML chapter. This chapter provided a good, although somewhat elaborate, example of how to use the DOM interface to parse a document in Java. While this was a good chapter, I was disappointed to see that the authors had chosen the Microsoft Java parser for their example. I ended up using the IBM's parser for my project because it had to run on multiple platforms. There are other good chapters (7 and 8) that explain the differences between HTML and XML, and how to display XML in a browser.

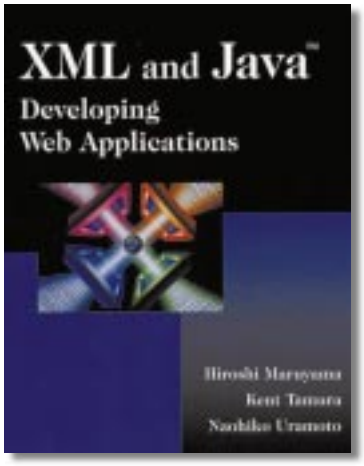
In conclusion, this is a good book for software engineers who are interested in the how and not so much the why of XML. My only real question to the authors is, "Why this fixation with Microsoft products?" I understand that Microsoft makes good products, but they only run on Windows platforms. What about the rest of the world?

—BY ISRAEL HILERIO

israel_hilerio@i2.com

XML and Java: Developing Web Applications

by H. Maruyama, K. Tamura and N. Uramoto
List Price: $39.95
386 pp
Addison-Wesley Pub. Co.; ISBN: 0201485435



Sometimes it's all in the timing. I was writing an application and looking for a reference for creating XML definitions for a virtual store's transactional objects. Specifically, I was looking for a book that covered XML parsing at the Java servlet level. Now writing this wouldn't be too bad, but I share a trait with millions of software developers – we're all lazy. Most books I had didn't get into elaborate examples. I'd searched the Web and though there were bits and pieces of examples in disparate Web sites, I didn't find a single comprehensive source for my needs. Just as I was getting ready to write my own code, a review copy of this book showed up at my doorstep.

This is a great book. It gets straight to the point, doesn't repeat a lot of information available elsewhere and is packed with abundant code in sufficient detail. The authors have an informal style that is very easy to follow. Salient points are provided as bulleted lists, short and to the point. The authors don't waste time with long, tedious explanations of the code, which is well commented and self-explanatory.

However, this is not a book for Java beginners. And though it briefly describes some XML concepts, if you don't know anything about XML or

Fall Internet

Java, this is definitely not the book to start with. The authors define the audience right in the beginning by saying, "This book is not a primer or reference on XML or Java.…We assume that you have at least a basic understanding of both and some experience writing Java programs."

That said, the book doesn't jump into advanced XML applications abruptly. Chapters 1 and 2 provide a fast-paced but decent introduction to XML and its ability to provide standards for data formatting. If you're familiar with Java and some markup languages (even HTML), you should be able to follow the chapters quite easily. In addition to the introduction, Chapter 1 also provides a good analysis of the synergy between Java, XML and the Web. Four application areas of XML are identified – document markup, metacontent, databases and messaging. These comprise the later chapters of the book. Chapter 2 covers XML parsing in just the right amount of detail for experienced programmers. This includes discussion on DOM and SAX interfaces. Since the authors are from IBM, the tool used for parsing is IBM's XML4J parser. In that sense this book is biased. However, the concepts should be applicable to other XML parsers for Java.

Chapters 3 and 4 get into more detail about XML documents. Chapter 3 discusses constructing XML documents, covering details on creating them from scratch as well as building valid DOM trees. The authors present a complete program for validating generation using DTDs. Readers not interested in such a deep level of detail should proceed to Chapter 4. Next, the authors provide techniques and examples for generating XML documents from DOM trees. Chapter 4 dives into the manipulation of DOM structures. The text is well broken out into method descriptions with appropriate illustrations. The chapter ends with a complex example, which develops an LMX processor for converting between XML documents. I didn't go through the example in detail as I don't have a current use for this application, but the discussion preceding it seems to indicate that the authors have done a great job in providing a sophisticated example. I haven't seen this level of detail in other XML texts.

Chapters 5–8 get more into enterprise-level programming with XML and Java. Chapter 5 covers how Java servlets can be used in conjunction with XML. A really good thing here is that the authors cover both the HTTP GET as well as POST methods for invoking the servlet. For XML, POST is more appropriate because of the length of the packet being transported. Unfortunately, POST is also the tougher nut to crack. The authors provide detailed explanations and corresponding code for using both mechanisms in servlets. A large part of the chapter consists of developing an XML-based Web document management system that uses servlets as the server-side processors of XML documents. There is well-commented code for serious developers trying to tie in pieces of the client and server side of XML.

Basically, Chapters 5–8 provide coverage on the server and middleware tiers of distributed computing with regard to XML. I haven't seen similar coverage in any other text. Chapter 6 covers JDBC and database access. A brief primer on JDBC is offered in the beginning of the chapter, followed by detailed descriptions of SQLX, a program that ties in the worlds of RDBMS and XML. Again, there is code aplenty for us coding fanatics.

Chapter 7 goes over messaging and security. If you've made it this far, you'll be expecting a primer on messaging and related technologies. The authors don't disappoint. The first half of the chapter covers messaging concepts, followed by a "PowerWarning" example that illustrates XML/HTML-based messaging. Later sections of the chapter cover XML message design and security using SSL.

The last chapter covers designing software components using XML and JavaBeans in conjunction. Some "XML Beans" are developed, and the chapter ends with a "Travel Planning Application" – a Web automation application using XML and JavaBeans. The Appendix describes other XML parsers and the XML for Java API reference.

This is a great book for developers trying to grapple with the different aspects of XML as they relate to distributed Java computing. I recommend this highly for advanced developers.

—BY TIJA RAGAS

tija@sys-con.com

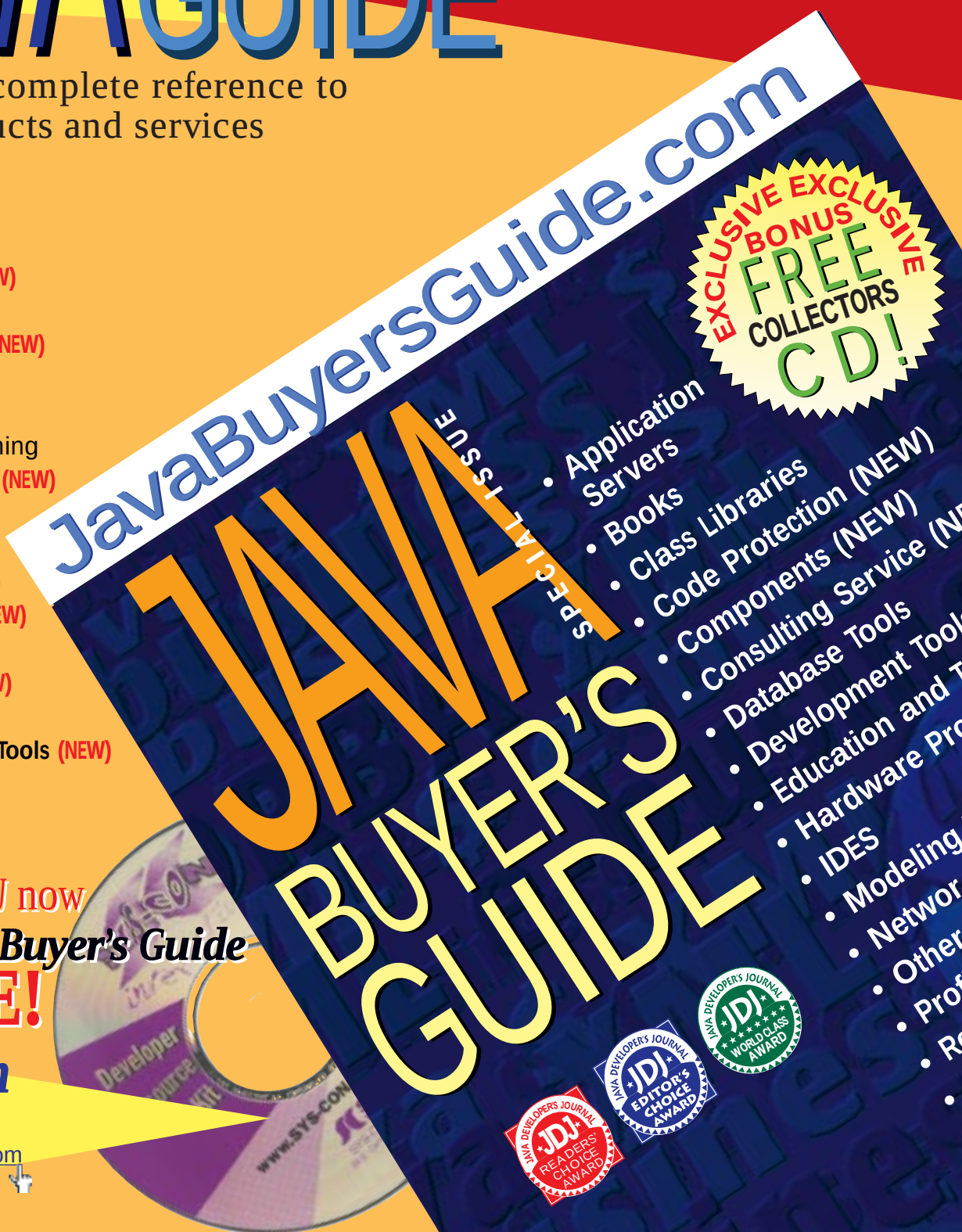
JAVA DEVELOPER'S JOURNAL presents

the Annual Print Edition of

JAVA BUYER'S GUIDE

the most complete reference to
Java products and services

- Application Servers
- **Books**
- Class Libraries
- **Code Protection** (NEW)
- Components (NEW)
- **Consulting Service** (NEW)
- Database Tools
- **Development Tools**
- Education and Training
- **Hardware Products** (NEW)
- IDES
- **Modeling Tools**
- Network Tools (NEW)
- **Other Java Tools** (NEW)
- Profilers (NEW)
- **Reporting Tools** (NEW)
- Sites (NEW)
- **Team Development Tools** (NEW)
- Testing Tools
- **Web Tools**



Subscribe to *JDJ* now
& receive ***Java Buyer's Guide***
FREE!

Attention

Java vendors!
update your listings at
JavaBuyersGuide.com
for our next print edition.

XML: The Next Generation EDI?

The vocabularies and dictionaries of electronic business communications

WRITTEN BY
KANG LU

ML has been touted to the point where you may think it's the greatest thing since sliced bread. As a technology, it's certainly achieved a level of respectable hype. However, does XML have what it takes to solve the problem of seamless business-to-business communication? To answer this question, I'd like to present XML under the limelight of EDI (Electronic Data Interchange). Using human-to-human communication requirements, I'd like to extrapolate those into business-to-business communication requirements and show that XML, by itself, solves only a portion of the problem but can act as an enabler for solving the rest.

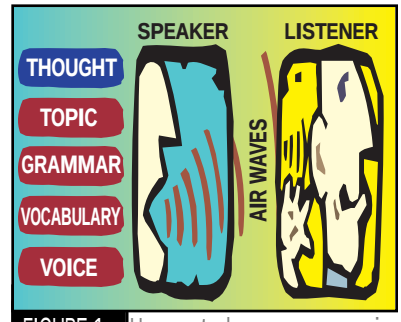


FIGURE 1: Human-to-human communication

How Do We Communicate?

I'd like to focus on the topic of human-to-human communication. This is an analogy similar to the one that RosettaNet, an organization geared toward specifying the interfaces for electronic commerce, used in their overview. It's a scenario we've all experienced and can easily relate to when compared to a similar machine-to-machine scenario. In Figure 1, when a person wants to talk to another person, he or she needs to develop a thought, usually containing some idea or topic the recipient will understand. The speaker must translate these thoughts into spoken words. This isn't a simple task. The speaker must identify the appropriate words from his or her vocabulary and group them according to some grammatical rules. The speaker then verbalizes a sentence or a phrase that produces a series of airwaves that is ultimately received by the listener. The listener hears the sentence and, using the same rules, processes it into a syntactic structure of recognizable words, interprets the semantic meaning and eventually duplicates the speaker's thought.

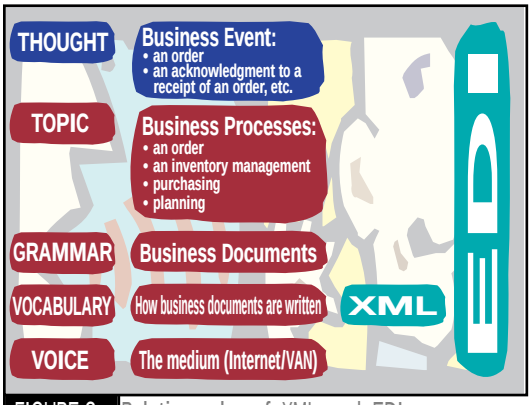


FIGURE 2: Relative roles of XML and EDI

We probably perform this simple communication task a thousand times a day. All of us take these simple acts of speaking and listening for granted. What we don't consider is the standard infrastructure in place that allows us to make verbal communication. What are these standards? The green boxes in Figure 1 represent an implied standard. If speakers and listeners don't have agreements on all the green boxes, communication will be impossible. For example, if the speaker is addressing one topic and the listener expects to hear a different one, the listener won't be able to put the speaker's sentences into context. I'm sure all of us have heard the phrase "What are you talking about?" Similarly, if the participants have different sets of vocabulary and grammar, the meaning of the original thought will be lost.

The Role of XML

In the growing world of electronic commerce, common standards are needed, especially in an environment in which machines talk to machines. Figure 2 illustrates where EDI and XML fit in the world of standards. You can see that the role of XML is smaller than that of EDI. Before you protest, allow me to explain. XML is a standard that governs how data is to be represented. It has provisions for defining data elements plus extending and customizing data structures. The power of XML is its flexibility to define an arbitrary data structure for both machines and people. A data structure defined in XML can be validated and easily parsed by using readily accessible, standard parsers. Its standards are predominantly applied to data schemas. What ASCII is to character encoding, XML is to data structures. In summary, XML gives you the power to specify the vocabulary but it doesn't impose or attempt to standardize the individual elements of the vocabulary. XML isn't a dictionary, only a means of formulating one. Using the example of human communication, you should be able to see that XML considered as a standard is simply not enough for meaningful communication. This is why, in Figure 2, the XML covers only part, not all, of the vocabulary section.

The Role of EDI

What is required is a standard that can embody a business dictionary in the context of business processes. This is where EDI comes in. With its long history, EDI provides common meaning to business transactions through the use of standard business documents. These documents – purchase orders, invoices, etc. – represent the contents of day-to-day business transactions. The set of business documents encapsulates business events and processes. Typically, a three-digit number is used to represent the document type. In the case of a purchase order, the number 850 is used. There are many standard bodies associated with EDI. The common ones are EDIFACT (EDI For Administration, Commerce and Transport), used primarily in international circles, and ANSI X.12, widely used in North America. EDI embodies more than just data definition, as it also dictates the transporting and administering of the documents.

A traditional EDI player is probably also a member of a Value Added Network. A VAN is like a private Internet hooking up businesses through modems. In addition to the network services, a VAN will also provide services along the lines of guarantee receipt, reporting, auditing and security. It's not

OMG

www.omg.org

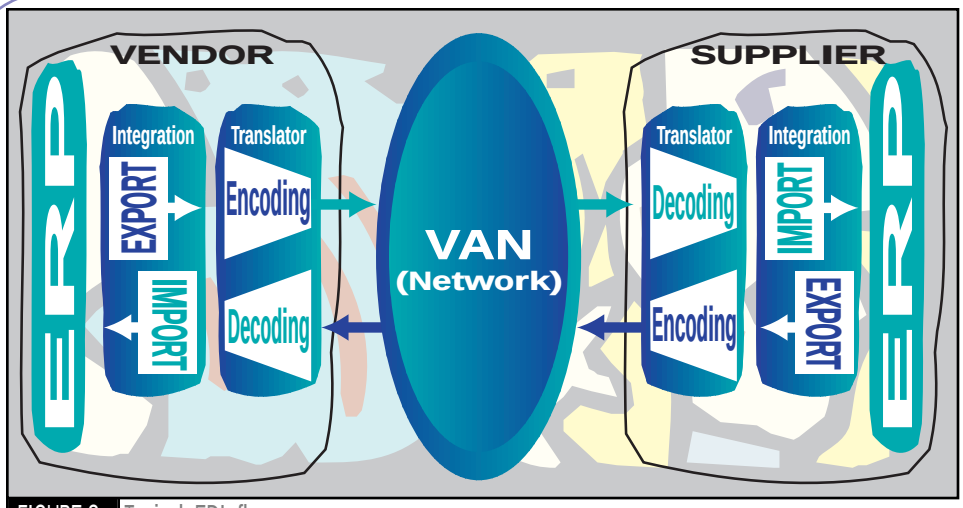


FIGURE 3: Typical EDI flow

cheap to obtain a membership in a VAN, which is generally more expensive than the Internet. In short, the VAN creates a virtual business community by acting as a hub for businesses to communicate using the EDI document standards.

Figure 3 shows a typical EDI transaction. A vendor creates an order document by populating fields in the document with information derived from its back-office systems, typically some form of ERP (Enterprise Resource Planning) system. Integration is required to extract the relevant values from this system. The values are then stored in the appropriate fields within the document via a piece of EDI software known as a Translator. The order document is sent along the VAN to be received by the supplier. The supplier, using a similar Translator, strips the relevant information from the order document and transfers it to its back-office system. During this transaction the VAN can provide various security and audit controls; it can also provide an acknowledgment to the vendor that the supplier has received the order. The supplier processes the order and in so doing sends an invoice document back to the vendor in a similar manner.

While the EDI documents provide a rich platform for business communication across different industries, from retail to government, they're sometimes viewed as being too rigid for certain businesses. Although the EDI documents provide optional capabilities, the documents can't be easily extended like XML. As business processes change, the demand for new fields arises. Users of EDI begin to tweak the standards by placing information where it doesn't belong in the documents. Data that didn't fit often ends up in unused fields. The semantic meaning of the fields then deteriorates, because now it represents information other than its original specification. This ad hoc solution works as long as all the trading partners are aware of the tweaks.

The concept of EDI is a good one. However, its implementation of a rigid document set sent over a proprietary network infrastructure – and expensive translator software for back-office integration – tends to inhibit the participation of new players in the world of electronic commerce. The XML technology opens the door for companies that didn't have the opportunity to participate in EDI in the past.

XML and Industry Standards

The XML community is offering a more effective solution. First, the coverage area of the VAN is a far cry from the coverage area provided by the Internet. The Internet also lessens the need for a proprietary network like VAN. Second, publicly available XML parsers are making it extremely easy to read and integrate with XML documents. Although this solves many technical issues of communication, the major part of the problem is still defining a set of standard business definitions that would include common grammar and vocabulary. It's worth noting that there's an XML/EDI Group trying to marry the flexibility of the data definitions of XML with the business language and practices of EDI.

Other major forces, such as RosettaNet, CommerceNet and OAG (Open Applications Group), are also trying to leverage the extensibility nature of XML to fill the grammar and vocabulary gap. These are organizations with big players that are undertaking the daunting task of specifying the business dictionary using XML. Within these organizations you'll find EDI standard bodies such as ANSI ASC X.12 and UCC (Uniform Code Council). There are also companies, such as Ariba, that are defining cXML, a set of XML schemas for procurement and catalog operations. New XML schemas are being created every day, which creates a new problem. Which definition should I use? Which dialect should I speak?

To tackle part of this problem, organizations such as Microsoft's Biztalk.org and OASIS's XML.org are beginning to take shape. Their purpose is to act as a clearinghouse for XML documents by soliciting business partners to collectively create a vocabulary repository. Different XML schemas are published in these clearinghouses for others to consume and use. They represent a hub for the business community of business dictionaries specified in XML.

The EDI experience has taught the electronic-commerce community about the need to customize and extend. An order document in the auto industry will be different from one in the electronic industry. The health-care industry will have business processes substantially different from processes in the retail industry. Each vertical industry will demand its own set of business documents. A horizontal process, such as order entry, will be different for each industry, resulting in different business documents to be transacted. The combination of the expressive power of XML and the development of big organizations to produce common business dictionaries is finally beginning to bridge the gap of our communication stack.

Is This Enough?

In the age of the Internet, XML offers an opportunity to rethink the implementation of EDI but it can't replace it by itself. The standard business dictionaries stemming from standard organizations are crucial to the success of business-to-business communications. The extensibility of XML will ease the traditional EDI pain of being too rigid in definition. The pace of competition is extreme and this means businesses need to be as agile as ever. This competitive environment forces businesses to change and upgrade their processes frequently. These changes in processes may lead to changes in messages. How well will XML, along with the armies of standard bodies, keep pace? Only time will tell.

References

BizTalk: www.biztalk.org
XML.org: www.xml.org
RosettaNet: www.rosettanet.org
CommerceNet: www.commerce.net
Open Applications Group: www.openapplications.org
Ariba: www.ariba.com
Data Interchange Standards Association (DISA): www.disa.org

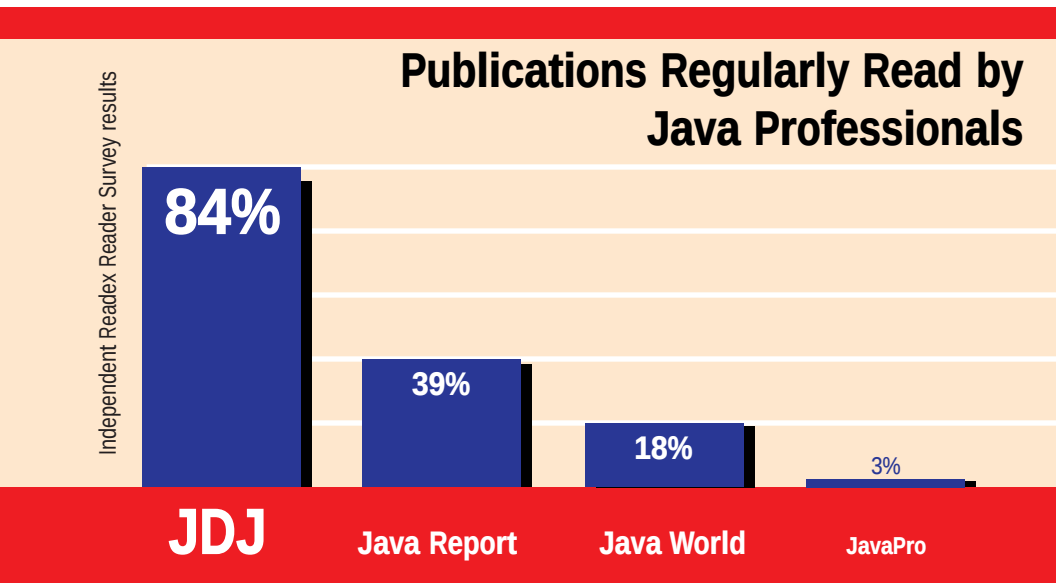
Author Bio

Kang Lu, a software development manager at Ironside Technologies Inc., Markham, Ontario, is a Sun Certified Java programmer with 11 years of programming experience. A graduate of the University of Toronto with a B.A.Sc degree, Kang Lu is responsible for the delivery of e-commerce–based architectures and infrastructures at Ironside.

klu@ironside.com

Only Java Developer's Journal Readers are

100% Pure Java



Before you advertise in a publication, please ask how many real Java readers you're actually reaching!

JDJ is the only publication whose readers are 100% pure Java developers.

Your ad in ***Java Developer's Journal*** reaches 100% Java professionals who make decisions to purchase Java related products and services, not over 40% Visual Basic programmers who never asked to receive the publication you advertise in!

We built our circulation one subscriber at a time.

That's one of our secrets why your ad works in ***JDJ.***

Carmen Gonzalez
Vice President,
JDJ Advertising Sales

www.JavaDevelopersJournal.com or call *914-735-0300*

The World's Leading Java Resource

RoboHELP Office 2000

by Blue Sky Software

Help... on Steroids

WRITTEN BY BERNIE METZGER

AUTHOR BIO

Bernhard Metzger is a Powerbuilder CPD Professional and president of KEV Systems, Inc., a consulting company in Newton, Massachusetts. Bernie can be reached at bmetzger@kevsys.com, or by voice at 800 376-5755.

bmetzger@kevsys.com

RoboHELP Office 2000: Blue Sky Software
Web: www.blue-sky.com
e-mail: information@blue-sky.com
Phone: 800 459-2356
International: 1 858 459-6365

Requirements:
32 bit OS running Windows 95, 98 or NT4.0 or later. Integrates with MS Word, supports Word 95, 97 or 2000

Pricing:
RoboHELP Office 2000 is scheduled to ship this month at a list price of $899

Help has arrived! I've have been using RoboHELP for about four years. I first bought it as a special at a Powersoft trade show in 1995. The price was right, it sounded like a cool product and I didn't know anything about creating help files. I create help files, mostly as technical help documents for the programs I write for my clients, and needed something that was easy to use. It looked like a good place to start. It was, it turned out, a good place to stay.

When I was presented with the opportunity to write this review, I was a bit overwhelmed. This is a very broad product with more features than I could write about in the space I have. Over the past four years the people at Blue Sky have made continuous improvements in this product. With WinHELP 2000 they continued this progression. With eHelp and other seamless Internet links they seem to be closing in on Ted Nelson's Xanadu.

The Quick Tour

RoboHELP Office 2000 is a multifaceted tool that helps you build not only classical Windows Help files for Win3.1, 95/98 and NT, but also HTML help, JavaHelp and Printed documentation. RoboHELP 2000 includes support for the new JavaHelp 1.0 and an interesting Web extension to help called eHelp.

JavaHelp

RoboHELP Office 2000 now supports the creation of JavaHelp 1.0 files for use in your Java projects. Sun Microsystems recently released JavaHelp 1.0 as a platform-independent Help format for Java-based applications. As Sun evidently has no intention of releasing an authoring environment for creating this kind of help, RoboHELP provides an easy development path with the single-source process – as long as you have the JavaHelp 1.0 and JDK 1.2.2. Once you've created your help project, you can easily generate the JavaHelp with all the Java-based Help features such as Table of Contents, full-text search, dynamic index, navigation controls, file compilation, HTML content, hyperlinks and Java components.

WebHelp

WebHelp provides a cross-platform solution for HTML-based help. It generates Help for Windows 2000, NT, Windows 98, 95 and 3.1, Macintosh and UNIX. It also provides a Java applet for browsers that don't support Dynamic HTML.

eHelp

The merging of classic help and the Web arrives with eHelp. Until now, a company's offering had to consist of static help files. eHelp creates a seamless entrée into search engines and gives you the ability to set up chat rooms or FAQ areas for your users.

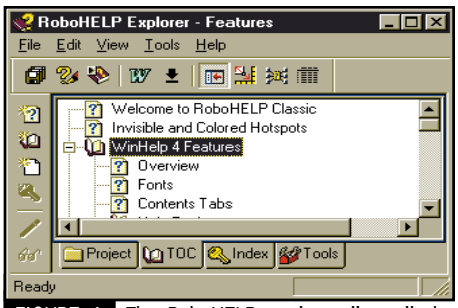


FIGURE 1 The RoboHELP explorer lists all the topics and components in your help project.

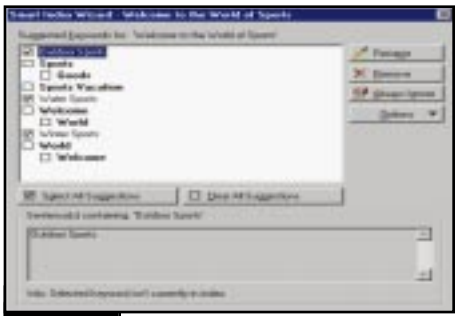


FIGURE 2 The Smart Index Wizard can automatically generate a comprehensive index.

FIGURE 3 A WinHelp system created with RoboHELP's WinHelp 2000 interface.

According to Blue Sky, "eHelp is a new Internet Help portal technology that Blue Sky is launching as part of RoboHELP Office 2000. eHelp extends a standard Help system by combining many of the Internet user-assistance tools into a single Help portal that can be accessed from within a standard application Help system. eHelp includes advanced Web search capabilities, and can also host separate communities for each application that employs eHelp. Each community will include a continually growing Q&A knowledge base and discussion groups, and allow real-time chat with other users of the same product." Blue Sky will host the eHelp on its own set of servers as a free service to users of RoboHELP. Contact Blue Sky if you intend to use this feature.

Getting Started with RoboHELP

You can create a new help project using RoboHELP's new project wizard. This wizard helps you build basic standalone help projects, application help with settings preset for many of the standard application environments such as Visual C++,

Access, PowerBuilder, Delphi and Visual Basic, to mention just a few. You can create help that presents itself in an Explorer pane and, if you have your own custom templates, custom help.

Once you've created the project, creating basic Windows help is easy. This is a very intuitive product. I actually have a manual somewhere, but I haven't opened it in a very long time. After you start up RoboHELP you have the RoboHELP Explorer on the left side of your screen and Word on the right (see Figure 1). To start, you create the outline for your help document as a series of books, and within them you create the topics you wish to write about (this is the CNT file). When you create a topic, RoboHELP adds the topic to the opened Word document and you then type the documentation. You can even generate a complete index automatically using the Smart Index Wizard (see Figure 2). When you finish, you click on the "compile" icon and poof! Help has arrived! While you can generate classic WinHelp output, you can also create a Help file that displays the TOC, index, etc. on the left side (see Figure 3) with the help of a DLL from Blue Sky.

Topic Options

RoboHELP really shines when it comes to the options available to you in the document itself. You can create hypertext links of amazing variety. First, you can create simple jumps to other topics within the same or other help documents. You can also add topics to a list of available "see also"s. Then, when you're creating another related topic, you can reference additional topics with a see-also button.

Buttons

RoboHELP provides four types of buttons you can place within help topics. These four types – authorable, mini, shortcut and graphical – determine the appearance of the buttons in your topic. What you can do with these buttons is quite extensive. You can define the display popups and jump to other topics within the same document, other documents or HTML pages. You can also provide a number of programmatic actions with the Macro button type. This button type lets you define menu items, keyboard shortcuts, execute programs, display video, play sounds and many other actions

Multimedia

Sometimes a picture, especially a moving one, is worth a thousand words. RoboHELP comes with a "Software Video Camera" with which you can capture both audio and video software demonstrations that you can include in your help files. You start your application and the "camera" as you point and click and type. The camera captures your actions and voice in an AVI file. You can then include this file in your help file for demonstrations.

PC HelpDesk

This feature combines database access to a knowledge base through your help files. PC HelpDesk packages your product knowledge base into a customized HelpDesk that users can access through a link from the help file.

Single-Source Output

One of the most powerful things RoboHELP provides is single-source output for help files. From one help project you can create output for a number of target platforms. One of my clients asked me if I could provide HTML help for the document I was writing. This is one of those requests that's easy to do with RoboHELP – and makes you look good! Once you've created the help project and done your documentation, you can create:

- Help for Win 3.1
- Help for Win 95 and NT
- Microsoft HTML help (Win 98)
- WebHelp3
- JavaHelp 1.0
- Windows CE Help
- Netscape NetHelp 1
- Netscape NetHelp 2
- Printed Documentation

The Real Power

With RoboHELP Office 2000, the power of an integrated world really becomes a possibility. With this new version you can integrate HTML into your help document. This means you can display HTML documents in your help file without exiting to a browser. You can jump from HTML to a normal topic, and you can jump to a Web page from your document.

OASIS Introduces XML Conformance Test Suite

(Boston, MA) – The Organization for the Advancement of Structured Information Standards (OASIS) announces the availability of the OASIS XML Conformance Test Suite – a set of more than a thousand tests that determine the ability of XML parsers to handle test cases built on the W3C recommendation. The suite incorporates tests developed by the OASIS technical committee with those contributed by Sun Microsystems, Fuji Xerox Information Systems and James Clark.

The suite can be downloaded from the OASIS Web site. www.oasis-open.org/committees/xmltest/testsuite.htm

OASIS Opens Membership to Individuals and XML Industry Groups

(Boston, MA) – OASIS has expanded access to its XML interoperability technical work by restructuring its membership requirements. The nonprofit consortium, previously open only to companies that use or provide products or services based on structured information standards such as XML, SGML and CGM, added new membership categories for individuals and associate XML industry groups. Interested parties may apply for membership online. www.oasis-open.org

Appeals Court Orders Reconsideration of Injunction Against Microsoft

(San Francisco, CA) – A federal appeals court ordered reconsideration of a judge's restrictions against shipments by Microsoft Corp. of software containing Java programming language.

U.S. District Judge Ronald Whyte of San Jose, California granted an injunction to Sun Microsystems Inc. last November. He said Sun was likely to show that Microsoft had violated a licensing agreement allowing it to use Sun's version of Java in its products.

The injunction prohibited Microsoft from distributing products that used Sun's Java copyrights unless Microsoft conformed to Sun's standards for Java.

The 9th U.S. Circuit Court of Appeals ruled that there was evidence to support Whyte's conclusion that Microsoft had violated the agreement by designing a version of Java incompatible with other software. But the court also said Whyte failed to explain why the alleged violation was a copyright infringement, rather than a breach of contract.

The distinction is important. A contract violation can justify an injunction against product shipments only if the innocent party can show it is being harmed. Otherwise, it must allow the shipments, then sue for damages.

aecXML Work Group Formed

A new aecXML working group has been formed to develop schemas for the exchange of AEC-specific business-to-business information. The aecXML schema work is being designed to assist software companies, construction firms, academic institutions, building product manufacturers and information publishers.

Bentley Systems has developed an initial specification for aecXML, a framework of XML-based schemas to facilitate communications related to designing, specifying, estimating, sourcing, installing and maintaining construction products and materials over the Internet. The working group is looking for interested parties to review and exchange ideas on the initial aecXML specification. www.aecXML.org

Scantron and V-Systems Form Alliance

(Tustin, CA) – Scantron Corporation and V-Systems, Inc. (VSI), have formed a strategic alliance to market Scantron FormTrap and VSI-FAX as a total enterprisewide document formatting and faxing solution.

The two applications will work in concert to intercept data from an organization's main database, format the data in a document file and then distribute via fax. FormTrap transfers the information requested from the database into designated fields within the document. While formatting the document, a user may edit text and add custom graphics to enhance its presentation. Once the document is complete, it resides on the printer server, or any network directory, ready to be distributed by the VSI-FAX system. VSI-FAX converts the print-ready output (document) to a fax format and distributes the document according to the embedded fax parameters. www.scantron.com and www.vsi.com

Java User's Group Formed in San Diego

(San Diego, CA) – The San Diego Java Users Group (JUG), a nonprofit organization founded in January 1999, meets once a month to address the hottest Java topics and breakthroughs. A typical meeting includes technical presentations and an open Q&A session. Past speakers include representatives from Sun, DTAI, Inc., Persistence Software, Symantec and Inprise.

The group meets at the Holiday Inn in San Diego's historical Old Town on the third Tuesday of each month. www.jug.dtai.com

Step 1 Acquires Unix Systems Administrators

(Kansas City, KS) – Step 1, Inc., Kansas City's premier provider of training to computer professionals, has acquired Unix Systems Administrators (USA), an Overland Park company providing UNIX operating system training.

Step 1's regularly scheduled one-week courses range from the fundamentals of C++ to advanced topics in Java, Visual Basic, Delphi, and object-oriented analysis and design. With the acquisition of USA, Step 1 becomes the only SCO-authorized UNIX trainer in a five-state region.

Step 1 courses are taught at their state-of-the-art training facility in Overland Park, Kansas. New classes form every month. www.step1inc.com

Internet Access Methods Introduces New Java App and Computer Training Courses

(San Francisco, CA) – IAM There, a new distance presentation tool with a call center server, is now available from Internet Access Methods. The Java-based technology provides the ability to create self-modifying dynamic content and integrate access to support from live expert help. www.iam-there.com

In related news IAM Consulting has released a new distance training format of courses and lectures by Java expert Gerry Seidman. Using IAM There technology, the multimedia presentations combine immediate phone access to expert help with the flexibility of offline computer-based training. www.iam-training.com

Bob Sutor Radio Interview *(continued from page 46)*

Sutor: XML.org isn't going to concern itself initially with looking at Java issues. It will keep itself to XML and work on the industry standard. That is already a big enough problem. Getting back to the first question, Java and XML are complementary. There are a lot of good tools for using them together. If you go to IBM's alphaWorks site, there are now 26 technologies available for doing XML, most of them in Java. Basically, Java is a great tool for dealing with XML.

***Sagar:* Why is that?**

Sutor: You want to use them in pretty much the same sort of environment. The same things that make Java so attractive: the independence of platforms and the interoperability.

***Sagar:* Let's talk about the organization of OASIS. There are different industries that are involved in it. How do you make sure it stays unbiased and doesn't have more representation from one industry to the other? I'm not talking specifically about IBM. I'm just talking about whoever gets in. What's the way you define committee selections and so forth?**

Sutor: We're still working on the process. In fact, we're looking at what happens in other organizations and we'll be building our process from that. You also have to remember that XML.org is part of a much larger OASIS, which already has about 90 numbers. We already have a lot of input. XML.org is not independent of OASIS, but rather an activity within it. So in addition to the partners in XML.org, we've got companies in OASIS such as Microsoft that have not signed up for XML.org, but they can comment on the standards process we are developing. Of course, we would welcome their support for XML.org and their joining the XML.org steering committee.

***Sagar:* What is the exact relationship between OASIS and XML.org? Is XML.org just a Web site for OASIS or is it a totally separate entity?**

Sutor: No, it's not a separate entity at all. We refer to it as an initiative, which means a project that is going on within OASIS. So today, basically, it's an organization that operates within OASIS. It doesn't have separate bylaws, so ultimately the OASIS board of directors makes the final decisions and has the fiscal responsibility. XML.org was formed initially by nine partners who put up $100,000 (large companies) or $25,000 (small companies) each to get this thing started. It's much more specific than the general OASIS. These people have helped to get it kick-started and so we can build the necessary infrastructure.

***JDJ:* I'm assuming we can go to XML.org to learn more.**

Sutor: That's correct – and every day the site will grow to become a real portal for learning more about using XML in industry.

ADVERTISING INDEX

While WORA May Be True, It's Still Half the Battle

WRITTEN BY ETHAN HENRY

"Write once, run anywhere" is probably the single-most repeated description of what Java is supposed to be about. It has been one of the cornerstones of Java's massive edifice of hype. However, like all hype, there's both truth and fiction to WORA.

The truth is that Java offers true cross-platform binary compatibility via the Java Virtual Machine and a rich set of standardized class libraries. Java has provided the ability to take an application from one platform to another without so much as recompilation like almost no other language or programming environment.

The unfortunate reality of WORA is that without a JVM and a set of associated libraries (which contain platform-specific native code), your Java bytecode isn't going anywhere. This is partly offset, of course, by the fact that for most operating systems, the OS vendors are quite willing and able to supply at least one, if not more, different JVMs for their platform. For example, at least four different JDKs are available for Windows (one each from Microsoft and IBM and two from Sun) and multiple JVMs from multiple companies. But if you're using something off the beaten track, like BeOS, you're out of luck.

The other unfortunate side effect of the WORA hype is Java's ill-fated inclusion in popular Web browsers, namely Netscape Navigator and Internet Explorer. It's true that Java's rapid explosion in popularity is due largely to the wide exposure Java obtained by being bundled into the one piece of software that just about everyone has on their computer, the Web browser. Unfortunately, as Java's abilities have grown through new features and new libraries (like Swing and Java2D), the ability to deploy real Java applications via browsers hasn't kept up. Even more unfortunately (and in my humble opinion here lies the real problem), this fact is virtually unknown to a lot of Java programmers, developers, architects and development managers. Somehow the true power of Java's write once, run anywhere abilities has become misunderstood to mean "we can ignore deployment problems – just run it as an applet!" In case I'm not being clear enough here, let me make it clear – applets don't work. Not for real applications.

The true power of Java becomes apparent when you look at real distributed application architectures using toolkits like Jini and JavaSpaces. JavaSpaces lets applications post not only data to a space shared by multiple distributed systems, but also code – it's a generic system that allows clients and servers (if those terms even mean anything in the context of applications written using JavaSpaces) to dynamically request just about anything. Jini is more than just a set of Java APIs, it's a vision of connecting devices together, and letting them discover each other independently and share code and data seamlessly. JavaSpaces itself is built on top of parts of Jini. In a recent interview Bill Joy, chief scientist of Sun Microsystems, admits that the concept of mobile code and a vision of Jini were in their minds when Java was originally directed onto the Web, back in '94 and '95. The concept of applets in this context is a brilliant marketing move, exposing everyone to Java and generating a huge amount of excitement. But realistically, without a mechanism for caching code and doing version management, you can't write a real application using applets. If you have a high-speed network and a small number of users, sure, they won't mind downloading a few hundred kilobytes of code (or more) every time they want to run the application. But once you have a serious number of users, a realistic, overstressed LAN and applications that are moving into the range of megabytes in size, the applet model just breaks down. Not to mention the incompatibilities between different browsers' Java implementations, making it difficult to do things like accessing the local disk or printing. And what about remote users, who are connected via low-speed links or who may spend significant periods of time completely disconnected from the network? There's a long list of problems.

None of this would matter much if no one was trying to use applets. But people are. All too often I'll be talking with a Java developer or reading a post on one of the Java newsgroups or mailing lists and I'll hear (or read) something that starts like this: "I'm having trouble with my applet. It's 947 Kb and it doesn't appear to be printing the Swing components correctly under Internet Explorer…." There's probably a solution for this developer's problem, but I think it's more than just another problem – it's a symptom. The fact is, it's hard to deploy software. Getting applications from the hands of the development team to the hands of hundreds, thousands or tens of thousands of users is a really hard task. Which is why so many developers cling to applets in the face of so many problems – they're looking for anything to help them with the problem of getting applications out and kept up to date on a regular basis.

The real solution is to use a more tried-and-true deployment technology, like automatic installers, or a more sophisticated enterprise-scale application deployment solution. Unfortunately, just as deployment is often the last thing that developers think about, it's one of the last major issues that has yet to be solved for the Java community at large. "Write once, run anywhere" may be true, but getting there is still half the battle.

AUTHOR BIO

Ethan Henry, KL Group's Java evangelist, can be reached at egh@klgroup.com.

egh@klgroup.com