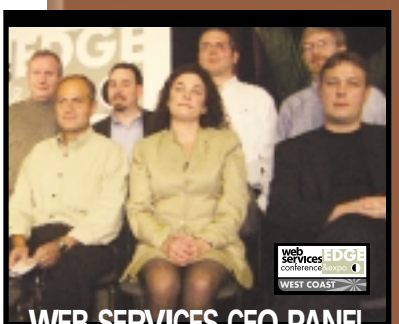
JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

November 2001 Volume: 6 Issue: 11

JAVADEVELOPERSJOURNAL.COM



WEB SERVICES CEO PANEL

AT WWW.SYS-CON.COM

PANEL PARTICIPANTS: ANNRAI O'TOOLE OF CAPE CLEAR; SEAN RHODY OF ***WEB SERVICES JOURNAL***; DIRK SLAMA OF SHINKA TECHNOLOGIES; GREG O'CONNOR OF SONIC SOFTWARE; ALI KUTAY OF ALTOWEB; EILEEN RICHARDSON OF INFRAVIO; AND BARRY MORRIS OF IONA

RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2002

$5.99US $6.99CAN

12

0 09281 01751 6

SYS-CON MEDIA

JAVA DEVELOPER'S JOURNAL

INTERNATIONAL ADVISORY BOARD

• CALVIN AUSTIN *(Lead Software Engineer, J2SE Linux Project, Sun Microsystems)*, • JAMES DUNCAN DAVIDSON *(JavaServlet API/XMP API, Sun Microsystems)*, • JASON HUNTER *(Senior Technologist, CollabNet)*, • JON S. STEVENS *(Apache Software Foundation)*, • RICK ROSS *(President, JavaLobby)*, • BILL ROTH *(Group Product Manager, Sun Microsystems)*, • BILL WILLETT *(CEO, Programmer's Paradise)* • BLAIR WYMAN *(Chief Software Architect IBM Rochester)*

EDITORIAL

EDITOR-IN-CHIEF: ALAN WILLIAMSON
EDITORIAL DIRECTOR: JEREMY GEELAN
ASSOCIATE ART DIRECTOR: LOUIS F. CUFFARI
J2EE EDITOR: AJIT SAGAR
J2ME EDITOR: JASON BRIGGS
PRODUCT REVIEW EDITOR: JIM MILBERY
FOUNDING EDITOR: SEAN RHODY

PRODUCTION

VICE PRESIDENT, PRODUCTION AND DESIGN: JIM MORGAN
EXECUTIVE EDITOR: M'LOU PINKHAM
MANAGING EDITOR: CHERYL VAN SISE
EDITOR: NANCY VALENTINE
ASSOCIATE EDITORS: JAMIE MATUSOW
GAIL SCHULTZ
BRENDA GREENE
ASSISTANT EDITOR: LIN GOETZ
TECHNICAL EDITOR: BAHADIR KARUV, PH.D.

WRITERS IN THIS ISSUE

CHARLES AREHART, BILL BALOGLU, CARL BARRATT, VINCENT BONFANTI, JASON BRIGGS, GLENN COATES, NEAL FORD, ABRAHAM KANG, TERESA LAU, DONALD LEVY, JAMES MCGOVERN, JIM MILBERY, ASHUTOSH NARHARI, BILL PALMIERI, MARK QUINN, AJIT SAAR, JULIEN TRANCHANT, ALAN WILLIAMSON, BLAIR WYMAN, JOHN ZUKOWSKI

SUBSCRIPTIONS:

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS, PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT

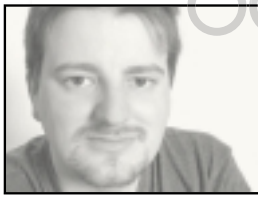
SUBSCRIPTION HOTLINE: SUBSCRIBE@SYS-CON.COM

COVER PRICE: $5.99/ISSUE
DOMESTIC: $49.99/YR. (12 ISSUES)
CANADA/MEXICO: $79.99/YR. OVERSEAS: $99.99/YR.
(U.S. BANKS OR MONEY ORDERS). BACK ISSUES: $10/EA., INTERNATIONAL $15/EA.

EDITORIAL OFFICES:

SYS-CON MEDIA 135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9600

JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is published monthly (12 times a year) for $49.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. POSTMASTER: Send address changes to: JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.



ALAN WILLIAMSON EDITOR-IN-CHIEF

<Web Services & XML>

This month the whole ***JDJ*** crew, minus Mr. Briggs, was at the Hilton in central Manhattan for our **JDJEdge** conference. It was a great show with much discussion regarding the state of our industry and where it's heading.

As usual the old "Web services" debate was high on everyone's agenda, discussing what exactly it means for us. I'm not too excited about the whole Web services revolution. Personally, I think it's just a marketing gimmick to repackage old products and technology.

Don't get me wrong, I'm all for opening up our systems, but Web services hasn't actually delivered anything new, merely packaged up the technologies we've already been using. I guess what it has managed to achieve is to bring "open systems" to the forefront and get people thinking about the best way to create them. So from that point of view I say, "Hurrah." However, I still cringe when I hear the phrase "Web services" mentioned in any conversation. As you can imagine, this standpoint doesn't make me too popular within **SYS-CON** circles, now that we have the ***Web Services Journal***.

Another debate that I managed to catch, which was very interesting, was the overuse of XML formats. This debate had some major heavyweights behind it; it was something that I've felt for a long time but for some reason I never heard it discussed. Maybe there were underground debates raging with people not wanting to go against the almighty XML revolution; I don't know.

Anyway, the underlying issue here is that XML is too verbose, inherently slow, and not suitable for all applications. Controversial I know, but an issue I fear that has to be addressed. The majority of uses I've seen for XML are on the whole a good use of the technology. It does the job well, communicating the metadata alongside the data so both parties know exactly what is going on. However, this comes at a heavy price, in terms of both transmitting and parsing.

Now that Java has gone mobile, coupled with the fact that the bandwidth is limited with the processing capabilities, we quickly see that XML is a little too bloated for mobile applications. I witnessed a number of wireless applications that were using an XML format to communicate between the server and the device. I asked if anyone else would be accessing the server, and was answered no, it was a closed system. So why on earth use XML as the format for passing data back and forth? As pretty as angular brackets are, think before you use them; they aren't always the best way forward.

There was an interesting press announcement at **JDJEdge** that piqued my interest. Merant announced that Microsoft had officially licensed its JDBC driver for use with SQL Server. It's a big deal, worth a lot of money for Merant, so congratulations there. However, looking at this from another angle, I can't help wondering what Microsoft is doing spending money on Java. Merant tells me that one of Microsoft's corporate clients asked Microsoft to provide Java support for its SQL Server database. I'd love to have been a fly on the wall in that conversation when Microsoft attempted to convince the mystery client to go against using Java and move to .NET. They must have been a fairly influential client to convince Microsoft to go against all its principles and officially support a language that it's desperately trying to kill off. More power to them, I say.

On that note I must bid farewell, but before I go, I want to tell you of a great discussion group that has survived the death of a column. **Straight Talking** is alive and well in the world of Yahoo groups (http://groups.yahoo.com/group/straight_talking_java). It originated in ***JDJ***, but it now has a life of its own. If you're up for some interesting and varied debate, check it out.

Catch you next month, and watch out for those <…> brackets!

alan@sys-con.com

AUTHOR BIO

Alan Williamson is editor-in-chief of Java Developer's Journal. *In his spare time he holds the post of chief technical officer at n-ary (consulting) Ltd (www.n-ary.com), one of the first companies in the UK to specialize in Java at the server side. Rumor has it he welcomes all suggestions and comments.*

WRITTEN BY VINCE BONFANTI

J2EE Without EJBs?

Did you use EJBs in your last J2EE project? Many Java programmers (and their managers and CIOs) would consider this a strange question. "How can it be a J2EE project if it doesn't include EJBs?" they might ask. The answer is: Sun currently lists 11 J2EE component technologies of which EJB is but one; of equal importance are servlets, JavaServer Pages (JSP), and JDBC. In fact, a recently released research report by Gartner, Inc., reveals that most Java projects do not use EJBs, but rely exclusively on servlets/JSP. (While not specifically mentioned in the Gartner report, I would guess that a high percentage of those projects also use JDBC.)

How is it that EJBs have become synonymous with J2EE in the minds of so many people? One answer is the natural tendency of software vendors to try to sell you the most expensive product. Is a commissioned salesperson going to sell you an inexpensive servlet/JSP solution if he or she can convince you to buy an "enterprise" application server (with EJB!) for 10 times as much? The tendencies of human nature should make the answer to that question fairly obvious.

Sun has contributed to the perception that J2EE requires EJBs through the J2EE licensing program and by not offering a separate certification program for servlets/JSP. The only way for a vendor to achieve Sun certification for a servlet/JSP implementation is by becoming a J2EE licensee. The costs of becoming a J2EE licensee are structured toward the "enterprise" vendors and are prohibitive for smaller companies. Therefore, you'll find the role of J2EE licensees dominated by EJB server vendors. Contrast this with JDBC (a J2EE technology), which has a separate certification program with a much more reasonable cost structure and a higher participation rate by smaller vendors.

Finally, we must not overlook the role of Java programmers, developers, and engineers in forming the equation that J2EE equals EJB. We (yes, I put myself in this group) naturally want to work with the latest, hottest, coolest, biggest, sexiest, most important new technologies. We sometimes overspecify and overdesign, and say things like, "Sure, we don't have a requirement for that now, but…." And we always have an eye on what will look good on our résumés.

Why is this bad? Because it creates waste. In the same study mentioned above, Gartner estimates that over $1 billion has been wasted since 1998 on purchases of EJB servers for projects in which EJB was not used at all. Instead, those projects were based entirely on servlets/JSP. Gartner projects that if this continues, another $2 billion will be wasted from 2001 through 2003. Interestingly, their numbers include only the cost of the application server purchase; they don't include the wasted engineering man-hours due to the added complexity of working with an "enterprise" application server, despite the fact that only limited use is made of the full functionality of these servers.

What does this mean to you and what should you do? Wasting $1 billion in the go-go dot-com and high-flying stock market era may not have been such a terrible thing. But wasting another $2 billion in the current economic climate is foolishness. Make sure you understand your project requirements. Yes, there are good and valid uses for EJBs, but there are many more projects for which servlets/JSP (and JDBC) are more than sufficient. Make sure you have a simple, inexpensive, easy-to-administer servlet/JSP application server in your technology arsenal. And the next time someone asks, "Is your project based on J2EE?" smile as you reply, "Yes, but we're not using EJBs."

vince@newatlanta.com

AUTHOR BIO

Vince Bonfanti is a cofounder and president of New Atlanta Communications. He has been a member of the Servlet and JSP Expert Groups since 1997.

Up and Running

I purchased a Handspring Platinum last week, and one of my first goals was to start writing Java apps on it ASAP.

After many failures, I dug through my copies of ***JDJ*** and found the issue with Jason Brigg's article on J2ME and application compiling ["Hello World: A Beginner's Guide to Writing Applications for the MID Profile," Vol. 6, issue 7].

Within a few minutes I had his HelloJDJMIDlet up and running on my handheld, and soon after I had one of my own applications running.

Many thanks for the great article.

Barry D. Ries
cnik2@home.com

Journeyman Column Works

Great article and welcome! I look forward to the new Arehart column. Charles Arehart has written a similar column in ***ColdFusion Developer's Journal*** that has been useful over the years, and I'm sure his **Journeyman** column in ***JDJ*** ["Making the Move to J2EE," Vol. 6, issue 9] will likewise encourage many people to delve deeper into J2EE.

Readers will save quite a lot of time and money by purchasing the books and perusing the other resources he recommends in his first column, and they will also learn J2EE better and faster. Charles always has carefully considered recommendations, and I always follow them to my great benefit.

Adam Phillip Churvis
info@ColdFusionTraining.com

Great article! "Making the Move to J2EE" is awesome [Vol. 6, issue 9]. I've been waiting for quite some time now for a "forget about the GUI stuff" review of Java books. Charles Arehart is taking the exact focus on Java that we need. I look forward to his next article.

Raymond Camden
jedimaster@macromedia.com

Where's the Code?

Congratulations! Page 72 of your September issue is a triumph of bad graphic design. It's not enough that you trash José María Barrera's article ["What Is Java Reflection?" Vol. 6, issue 9] by not including the code listings he refers to, but you then make a complete dog's dinner of what you do print.

Terry Child
terry@smallblueworld.com

Editor's Note: *The code listings the author refers to are available on the* **JDJ** *Web site, as cited in the article.*

Isn't It Ironic?

I read Alan Williamson's editorial ["Together We Stand, Divided We Fall," Vol. 6, issue 9] with some interest because it reminded me of how the Smalltalk vendors felt a few years ago when Sun flexed its mighty marketing muscles to place Java in the spotlight before it was ready, throttling the momentum that Smalltalk had finally begun to gather. Now Java has matured and, despite a few warts, rightly holds its place in the realm of development platforms.

I find it ironic that Java users now worry over the upcoming threat from Microsoft's marketing. One thing I've learned as a reformed Smalltalker now doing Java is that in the battle over which technology to adopt, it's rarely about the technology. So I agree with you that those of us who want to continue using and recommending Java will have to work extra hard to earn that right.

Michael Silverstein
msilverstein@silvermark.com

Print an Annual Index

I've been reading ***Java Developer's Journal*** for several years now, and it keeps getting better and better. I love the latest format and its organization.

One problem I encounter with my many stacks of publications and magazines is how difficult it is to find the exact article I'm looking for. While having the table of contents listed on the cover helps, it still means having to go through each issue to find something.

If I might make a suggestion, some publications, particularly professional journals, print an annual index cataloging all articles published in that year. I'm sure many other people would find such an index tremendously useful.

Thanks for publishing a great Java resource, and keep up the good work.

Bonnie SooHoo
bsh01@onebox.com

ERRATA

Due to a printer's error, the last page of Dan Malks article, "Core J2EE Patterns" (JDJ, Vol. 6, issue 10), did not appear in the print issue of the magazine. The entire article can be accessed by logging on to www.sys-con.com/java/article.cfm?id=1161.

JDJEDGE conference&expo™

JDJEdge 2001 We

The Show Goes On

written by Alan Williamson

OVER 4,200 DELEGATES REGISTERED TO ATTEND BEFORE SEPTEMBER 12

CONFERENCE ATTENDANCE EXCEEDS 2,700

JAMES GOSLING DELIVERS OPENING KEYNOTE TO MORE THAN 1,200 DELEGATES

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

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

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

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

Between September 23 and 26, **SYS-CON Events** hosted its ***JDJ***/Web Services East conference in the heart of Manhattan. In light of the tragic events a couple of weeks earlier, we made the agonizing decision to follow the advice of Mayor Giuliani and get back to normal as quickly as possible. With that, we had near full attendance from our exhibitors and a nominal no-show from some of our speakers, who understandably felt safer not flying.

David Litwack, CEO of SilverStream, delivering his keynote

An attentive audience filled the Hilton ballroom to hear James Gosling's keynote presentation

Web Services Journal editor-in-chief Sean Rhody opened up the conference to a packed hall and indicated the value that attendees would be getting in the next few days. This included a rich overview of Java and the prolific implementation and adoption of Web services. James Gosling presented the first keynote, offering insight into Java and where it was heading with respect to the whole Web services revolution.

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

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

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

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

b Services Edge 2001

web services EDGE conference&expo™

James Gosling, Ajit Sagar (XMLEdge 2001 conference tech-chair and editor-in-chief of ***XML-J***) and Alan Williamson (editor-in-chief of ***JDJ***) at the JDJEdge 2001 Expo floor

Dr. Richard Mark Soley, Chairman and CEO of Object Management Group, delivering his keynote speech at JDJEdge 2001

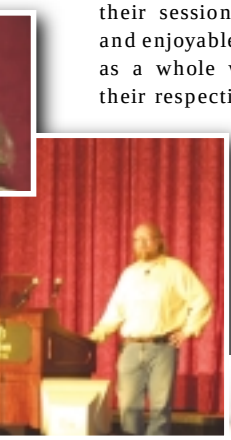
of two camps: those that can, and those that cannot. I've found that the ones that have captured my interest (and the room in which they speak) are those that have a coding background and still consider themselves developers. The likes of Dave Chappell from Sonic and Tyler Jewell from BEA are two great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

The second day opened up with a Web services panel of the industry's heavyweights, chaired by our own Sean Rhody. Don Leclair, James Gosling, Richard Soley, Tyler Jewell, Dave Chappell, Rick Ross, and Dave Litwack answered questions from both Sean and the floor. It was interesting to note that the panel spent a significant amount of time defining exactly what the term *Web services* connotes.

Greg Kiessling, CEO and Founder of Sitraka, presenting his session

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

> "I applaud the 'show must go on' attitude, both on the part of the attendees and SYS-CON. From the interactions I had with many of the attendees, I am very impressed with the high caliber and technical adeptness of the audience"
>
> —Dave Chappell, VP and SonicMQ Chief Technology Evangelist



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

Conference & Expo 2001

JDJEDGE conference&expo

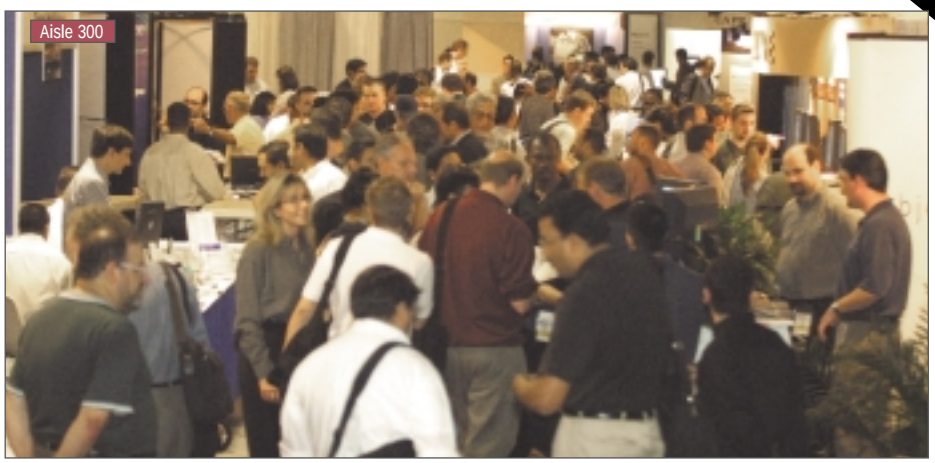
One thing the panel did agree on was the need for more open systems. Whether or not XML was the best tool for the job came up as well, with James Gosling and Richard Soley both playing the performance card – there is too much redundancy inherent in the protocol, which only serves to eat up valuable bandwidth and unnecessary parsing.

SavaJe announced the winner of its developer's competition that was run in conjunction with ***Java Developer's Journal***. The winner of the large LCD monitor was Passport for its Java Remote Presentation Applet. Full details on the winners can be found at www.savaje.com.

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

web services EDGE conference&expo

Exhibitors made numerous contacts at the Expo.



The largest gathering of Java professionals on the East Coast


KEYNOTE PANEL: WEB SERVICES PARADIGM

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

SYS-CON TV www.sys-con.com

Watch JDJEdge 2001 on SYS-CON Television at www.sys-con.com

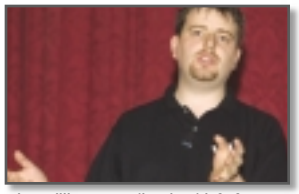
Conference & Expo 2001

JDJEDGE conference&expo

web services EDGE conference&expo

Joe Menard, chairman and CEO of WebGain, received a *JDJ* Readers' Choice Award from Alan Williamson

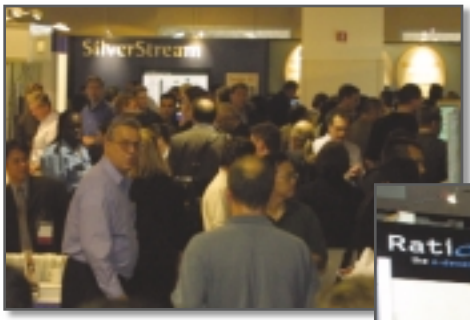
TIBCO Software explained their total business-integration solutions



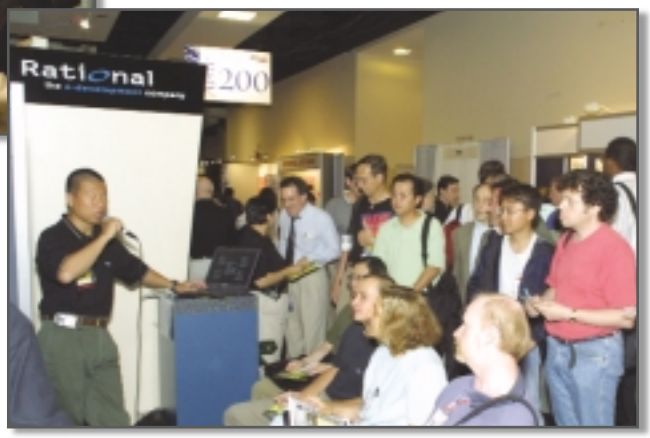
Alan Williamson, editor-in-chief of *JDJ*, introduced James Gosling at the opening keynote

Compuware demonstrated their latest products

Contacts are made at the Compuware booth



SilverStream Software introduced the eXtend product line at JDJEdge



Attendees lined up at the Rational Software booth to hear about Rational's integrated, full life-cycle solution

Scott Dietzen, CTO of BEA's e-commerce division, delivered his keynote…

Scott Dietzen, CTO of BEA's eCommerce Division, stopped for five minutes during his presentation to actually build and demonstrate a Web service. While trivial, it did show the power of the J2EE tools available already, and how they can be exposed as Web services with very little effort. Dietzen actually demonstrated service chaining, combining two Web services to obtain traffic information in French – one to obtain the information, another to translate it from English (à la Babelfish-esque technology). He pointed out how naturally Web services fit on top of the J2EE stack, and how the design principles already employed for J2EE design serve well in the Web services world.

…then answered questions from the conference delegates

Rick Friedman, founder of SIGS Publications and *Java Report*, visited the Expo floor with Fuat Kircaali, CEO of **SYS-CON Media**

Conference & Expo 2001

JDJEDGE conference&expo

web services EDGE conference&expo

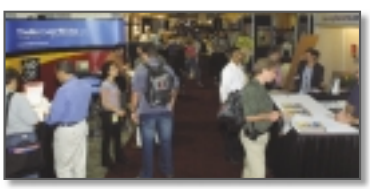
BEA demonstrated the latest version of WebLogic Server, which garnered top honors at the *JDJ* Readers' Choice Awards

Nextel demonstrated the latest in wireless communications

Cape Clear demonstrated tools aimed at simplifying Web service deployment

Attendees heard about Object Design's Java data caching and management solutions

Delegates get the latest information at the ParaSoft and ReportMill Software booths



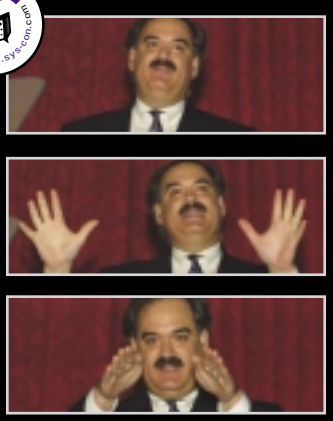
The ServerSide.com and Middleware Company discussed their newest products with conference attendees

There were a number of vendors present that were focused primarily on Java, but almost all of the vendors had some take on what a Web service was and how their product line did or would play a part in the world of Web services.

At the Axiomatic Design booth attendees discovered how to write perfect code

The exposition floor was packed with vendors exhibiting wares aimed at Web services. Cape Clear and Shinka were both present, demonstrating tools aimed at the simplification of Web service deployment. SilverStream demonstrated their eXtend product set, and has made a free Web services container available to developers on their Web site. BEA demonstrated the latest version of WebLogic Server, which has been tuned to provide Web service performance from the J2EE stack.

SYS-CON TV www.sys-con.com



RICHARD SOLEY
CHAIRMAN AND CEO OF OMG
DELIVERING HIS KEYNOTE SPEECH

"JDJEdge 2001 International Java Developer Conference & Expo was a tremendous success! Following Mayor Giuliani's advice in the wake of the September 11 tragedies, the SYS-CON organization made the gutsy call to go ahead with the show – and they proved that JDJEdge is an important industry event. A large, technically astute crowd took the opportunity to learn more about Web services and the Java platform. The high attendance proves the importance of this venue, and I'm glad I was able to come and share information on model-driven architecture, Java, and Web services."

Richard Soley
Chairman and CEO
OMG

Aligo demonstrated their M-1 Mobile Application Server

Conference & Expo 2001

JDJEDGE conference&expo

web services EDGE conference&expo

MERANT DataDirect announced the upcoming beta version of the MS-SQL Server 2000 JDBC driver

Attendees received a first-hand demo of e-business applications at the Compuware booth

SavaJe Technologies demonstrated the latest release of the SavaJe XE Java operating system

A discussion regarding CodeHound's Internet search engine for software developers

Conference attendees discussed the latest technology at the TIBCO booth

Attendees heard how Cape Clear Software's products automate the business Internet

Pingtel demonstrated their Java-based, voice-over-IP phone appliances, softphones, and management software

The CSC booth was open for business

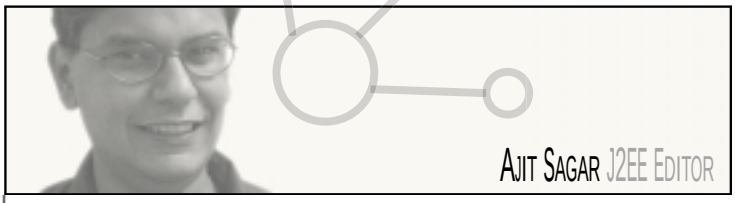
Discovery and Web Services UDDI was a topic on everyone's minds. The panel discussed the ability to create a global DNS as well as the real value in having one particular global system. The prevailing opinion seemed to be that UDDI was destined for smaller, private usage (i.e., a domain per industry) with only limited deployment of services to the public. Part of this was due again to the social nature of the way people do business – they like to know their trading partners. "The idea that Joe's tire shop will provide the next million tires to Ford just because they have a Web service for tires is ridiculous," was one of the more memorable quotes from the panel. People share with other people based on relationships, and any technology that ignores this is doomed.

SYS-CON TV www.sys-con.com

DAVE CHAPPELL
RENOWNED AUTHOR AND CHIEF TECHNOLOGY EVANGELIST OF SONIC SOFTWARE TEACHING TO A PACKED CLASSROOM AT JDJEDGE...

"JDJEdge 2001 International Java Developer Conference & Expo was very well attended. The mood of the attendees was remarkably upbeat, given the recent tragic events and the proximity of the show. I applaud the 'show must go on' attitude, both on the part of the attendees and SYS-CON. Also, speaking from the perspective of a speaker, and the interactions that I had with many of the attendees, I am very impressed with the high caliber and technical adeptness of the audience that was drawn to this conference."

Dave Chappell
VP and Chief Technology Evangelist
Sonic Software



AJIT SAGAR J2EE EDITOR

A Mix of Sun and Clouds

First things first: I'd like to take a moment to reflect upon **SYS-CON's JDJEdge Conference** in light of the tragedy that shocked America on September 11. The best way to deal with a terrorist act is to not give in and allow it to disrupt everyday activities. To that effect, JDJEdge took place as planned, September 23–26. Under the circumstances, it was quite successful.

On the J2EE front, most vendors attended and imparted useful information in the form of presentations, exhibits, demos, tutorials, and one-on-one discussions. Kudos to the SYS-CON crew, who worked under adverse circumstances to make this happen, as well as to the attendees, vendors, presenters, faculty, and industry luminaries who traveled from across the globe to help shape the event.

Sun announced the final release of J2EE 1.3 on September 24, the second day of the conference. Java versions can get confusing. This was the version 1.3 release of the API suite for the enterprise edition. The 1.3 release is promoted by Sun as the release that "simplifies business integration and delivers increased functionality for developing and deploying enterprise-level Web services." If you look for the message in this statement, it covers two main concepts: "business integration" and "Web services." Basically, these two paradigms are Java's windows into enterprise applications.

Web services are abstractions of functional modules built in a programming environment and applied to Web-based enterprise applications. The main effect of creating Web services is to have the functionality available in a form that can be queried, brokered, searched, and executed across the Web. On the other end of the spectrum, however, is the actual integration of the Web service to the back end. In a world replete with Web services, this will be achieved through the Web layer. For instance, business partners can connect via XML-based protocols talking Web service to Web service. The connectivity layer, however, isn't that simple yet, and chances are it will never get to that level. Integration to the current EIS systems can be done via JDBC (for database sources), JMS, CORBA, JNI (if absolutely required), or the J2EE Connector Architecture (J2EE CA). It's Sun's hope that the integration market will move toward the J2EE CA as the preferred way of integrating to legacy systems. J2EE CA provides the connectivity layer to complete the final piece of the enterprise application story.

The J2EE 1.3 version continues enhancing the Java platform to support enterprise application development. The main features of this release are:

- The release of J2EE connectors
- JMS message-driven beans
- Enterprise JavaBeans 2.0 release
- JAXP for XML integration and XML manipulation using JSPs
- Java servlet filters

As in the other releases, 1.3 comes with a reference implementation that developers can use to become familiar with the environment. As you can see from the preceding list, most items deal with enterprise connectivity and Web services.

As vendors that support the Java platform continue to enhance the application development and deployment environments, more sophisticated tools for building enterprise applications are emerging. We are, however, just on the brink of the e-business revolution.

In this issue we have an article from Neal Ford on Jakarta's Struts framework that applies the MVC design pattern to Java presentation design using servlets. Abraham Kang and Donald Levy introduce you to the joys of implementing single sign-on in the Java arena. A couple of product reviews and our regular columns complete the picture to give you another substantial bite of J2EE in true ***JDJ*** style.

ajit@sys-con.com

AUTHOR BIO

Ajit Sagar is the J2EE editor of JDJ and the founding editor and editor-in-chief of XML-Journal. A senior solutions architect with VerticalNet Solutions, based in San Francisco, he's well versed in Java, Web, and XML technologies.

J2EE INDEX

Security for J2EE Applications

Implementing a single sign-on

Part 1 of 2

WRITTEN BY
DONALD LEVY & ABRAHAM KANG

Most developers build J2EE applications using their own security mechanism. This causes problems when other applications are introduced, because more logons have to be remembered and users have to physically log on multiple times to use different Web applications. A single sign-on allows a user to logon once and have transparent access to all the applications within a domain.

In Part 1 we explain what a single sign-on is and how it works, enabling you to implement a single sign-on solution. In Part 2, we'll implement a single sign-on solution for standalone and Web-based applications.

Single sign-on (SSO) in client and Web applications share common behavior but also differ due to their content delivery mechanisms. In Web applications, single sign-on allows a user to log in once to a centralized server and transparently access multiple sites or Web applications without having to log in to each Web application. Single sign-on for a standalone application allows a single application to obtain services from multiple servers by reusing a set of credentials through the Java Authorization and Authentication Service (JAAS).

Single Sign-on for Web Applications

Single sign-on in Web applications uses secure cookies, redirection, encryption, life-cycle management servlets, member servers, and a login server.

Secure cookies are Web browser cookies that can't be copied, read, or sniffed. Normal cookies are usually written to a cookie file. This is inherently insecure because an attacker could copy this cookie information to another machine where he or she could masquerade as the victim. In-memory cookies are not persistently written to a cookies file but to the browser's memory space. As a result, in-memory cookies protect users from a cookie file attack.

Cookies that can't be read have their payload encrypted. Although a cookie payload is encrypted, an attacker could sniff a network segment to obtain the cookies with encrypted payloads and initiate a replay attack. A hacker uses a replay attack by sniffing for cookies set after a user logs in. Although the hacker can't read the information in the cookies, he or she can use the cookies to masquerade as the victim.

To protect against sniffing attacks use the HTTPS protocol when transferring login server cookies to and from the client. This ensures that any information sniffed will be useless. The login server uses secure cookies to keep track of whether or not a user is logged in.

Redirection is the process by which users are transparently sent from member servers to login servers and vice versa. Redirection can be accomplished with META tags or JavaScript.

Encryption is the process by which clear text is transferred into cipher (encrypted) text. Symmetric and asymmetric encryption are two forms of encryption. Symmetric encryption utilizes a single key to encrypt and decrypt information. Asymmetric encryption utilizes a key pair where anything encrypted by one key can only be decrypted by the other. It's possible to use either method with SSO; however, symmetric key encryption allows all member servers participating in the login domain to share secure information using a single key, which is easier to manage and requires less maintenance when replaced.

The *life-cycle management servlets* allow the login server and member servers to notify each other of login, registration, and logout events. Each member server and login server has an LCM servlet that serves as the sole communication point between a member server and a login server. LCM servlets are accessible only through SSL. The LCM servlets have slightly different roles depending on whether they're located on member servers or the login server. The LCM servlet on the login server receives requests from member servers to login, register, or logout users. After the login server logs in, registers, or logs out a user, it notifies member servers through their corresponding LCM servlet via a redirect.

Member servers delegate the authentication of users to the login server. They represent the different applications that need to be integrated under an SSO domain.

The *login server* is the central authentication service and is responsible for major events in a Web site–user's life cycle. The major events include registration, login (authentication), and logout. When a user needs to register and set up an account with any of the member servers, they're redirected to the login server.

The login server presents a common registration page that encompasses all information required by participating member servers. The login server stores this registration information, stores the member server details, sets a cookie, and redirects the user back to the LCM servlet of the member server that originated the request. All the encrypted registration information is sent back as well so the member server can store this information locally. The member server decrypts the information, then logs the user in and establishes its own session

management mechanism.

A similar process occurs when a user logs in. First the member server notices that a user has not logged in. The member server redirects the user to the LCM servlet on the login server with a login directive. The login server notices that the user has not logged in and presents a customized login page for that particular member server. Once authenticated, the login server stores the member server details, sets a cookie, and redirects the user back to the LCM servlet of the member server that originated the request. Encrypted user information is passed back to the member server, which then decrypts the information, logs the user in, and establishes its own session management mechanism.

When a user clicks a link to another member server (where the user has not been authenticated), that member server notices that the user has not been authenticated. The member server redirects the user to the LCM servlet on the login server with a login directive. This time the login server notices the login cookie, stores the member server details, and redirects the user back to the LCM servlet on the member server that originally redirected the user to the login server. Encrypted user information is passed back to the member server, which then decrypts the information, logs the user in, and establishes its own session management mechanism. This process is repeated for every other member server when accessing a particular member server for the first time after logging in to the login server.

When a user logs out of the application, none of the member applications should be left open. The user initiates a logout by clicking a logout icon. This redirects them to the login server's LCM servlet with a logout directive. The login server has stored information about which member servers the user has logged into and redirects users to each of the member server's LCM servlets with a logout directive. Again encrypted user information is passed to the member servers. Each member server redirects the client back to the login server so it can record the successful logout of a particular member server. It redirects the user to the next member server he or she needs to log out of; this process continues until there are no member servers left. At this point the login server returns a logout success page.

Now that you've gained a strong understanding of single sign-on for Web applications, let's look at how single sign-on is implemented for standalone applications.

Single Sign-on for Standalone Applications

Single sign-on for standalone applications uses JAAS, General Security Service API (GSS), and a centralized authentication resource. JAAS provides the authentication and authorization mechanism. The GSS API provides mutual authentication, confidentiality, and integrity of individual messages passed between a sender and a receiver over a stream. A centralized authentication resource provides ease of maintenance and a unified API for integrating Java applications with single sign-on. There are many resources on the Net that provide in-depth information on these technologies. The following is a brief introduction.

Java Authentication and Authorization Service

As the name implies JAAS is a framework that allows Java programs to authenticate and authorize users. Authentication revolves around the following classes: Subject, Principal, LoginModule, CallbackHandler, and LoginContext.

A *Subject* represents a client to an authentication service, and can be people or other applications. After it's authenticated it has a number of Principals associated with it.

Principals represent the unique identifiers for an authentication domain. An example of a principal in a database could be your employee ID. A principal in Kerberos is a string that looks like "abe@infogain.com". An IRS principal would probably be your social security number. LoginModules physically assign Principals to Subjects.

LoginModules abstract the general authentication mechanism of your application in a pluggable fashion. A simple example of a LoginModule would be one that authenticated your credentials using a database. A more complex LoginModule might require biometrics for authentication using a temperature and heart rate–sensitive fingerprint scanner. The actual interface to the authenticating hardware uses CallbackHandlers.

CallbackHandlers are responsible for interfacing with the authentication hardware to obtain the actual proof that a user is who he or she claims to be. A NameCallback and PasswordCallback have already been implemented to enable you to get the username and password from a user. Other CallbackHandlers can be written to interface with SmartCard readers. An application interfaces with JAAS through a LoginContext.

The *LoginContext* is the authenticating application's interface into JAAS. It's responsible for initiating the login (using the LoginContext.login method) and retrieving the authenticated Subject (using the LoginContext.getSubject method). The LoginContext's constructor takes two arguments. The first names a JAAS-specific policy file to select one or more LoginModules used for authentication. The second parameter identifies a CallbackHandler to retrieve authenticating credentials from the user. After a user has been authenticated, JAAS binds Principals to the user's Subject, which is used by JAAS Authorization.

JAAS Authorization controls user actions based upon the Principal objects associated with the authenticated user's Subject and policy file permissions. There are three main components to JAAS Authorization: Subject class, PrivilegedAction class, and a Java policy file.

JAAS Authorization uses the Subject.doAs(Subject, PrivilegedAction) and Subject.doAsPrivileged(Subject, PrivilegedAction, AccessControllerContext) methods to control code (in the run method of the PrivilegedAction) that has permissions associated with it. Subject.doAs( ) associates the specified Subject with the current Thread's AccessControlContext. Subject.doAsPrivileged( ) associates the specified Subject with an AccessControlContext passed into the method. The code that must abide by policy file rules is encapsulated within the run () method of the PrivilegedAction interface, which has one method (see Listing 1).

JAAS Authorization has extended the standard Java 2 policy file by adding a Principal section after the signedBy and codebase sections. For example:

```
grant signedBy "abe",
      codebase
"file:./someFile.jar",
      Principal com.infogain.secu-
```

> "Although the hacker can't read the information in the cookies, he or she can use the cookies to masquerade as the victim"

```
rity.principal.InfogainPrincipal
"joek" {

      permission
java.util.PropertyPermission
"java.home", "read";
}
```

Before JAAS Java could only restrict code actions based on the signer of the code and where the code existed on the file system or network. With JAAS you can restrict a specific user's actions. In the above grant block we're saying that the only principal allowed to read the "java.home" system property is a Subject that has an InfogainPrincipal storing the value "joek". Although some might deem JAAS sufficient for a single sign-on, I feel it's necessary to add additional security measures for data transferred between the client and server.

General Security Service API

The General Security Service API provides a way of securely transferring messages between a client and server utilizing a single sign-on infrastructure. It's an implementation of RFC 2743 and has its own RFC in 2853. According to RFC 2853, "The GSS-API allows a caller application to authenticate a principal identity, to delegate rights to a peer, and to apply security services such as confidentiality and integrity on a per-message basis." The two supported centralized security services are Simple Public Key Mechanism (SPKM) and Kerberos Version 5. In this article we focus on Kerberos V5.

Centralized Authentication Resource

All applications delegate authentication responsibility to Kerberos. Kerberos is a single sign-on service that acts as a mediator when users and services want to authenticate themselves to each other. It was developed at MIT based on work by Needham and Schroeder. It's available on almost all platforms and has been incorporated into Windows 2000/XP. Keberos provides a single management interface for user/password management. Kerberos also provides the underlying security services for mutual authentication, confidentiality, and integrity used by the GSS API.

When using JAAS and Kerberos, single sign-on becomes transparent to the application user. To utilize single sign-on, users start their applications, authenticate to JAAS, and access any distributed applications running on the network.

In Part 2 we'll show how to implement single sign-on using code examples.

Author Bios

Abraham Kang is an integration architect for Infogain's Enterprise Integration Solutions group. His focus is J2EE, security, and enterprise application integration. Abraham has been developing in Java for five years.

Donald Levy is the director of alumni relations and development information systems at Stanford University. Donald has a BS from MIT and an MS from Columbia University.

abraham_kang@yahoo.com

donald.levy@stanford.edu

Listing 1

```
   public Object run ();

Here is an example of a class implementing PrivilegedAction:

   public class TestAction implements java.security.PrivilegedAction {

       public Object run () {
  System.out.println("You are supposed to do some controlled " +
"action like getting a system property (user.home = " +
    System.getProperty("user.home") + ")";
 }

   }
```

Download the Code! www.JavaDevelopersJournal.com

M-1

Mobile Application Server

by Aligo, Inc.

REVIEWED BY JIM MILBERY
jmilbery@kuromaku.com

Aligo, Inc.
444 De Haro Street, Suite 211
San Francisco, CA 94107
Web: www.aligo.com
Phone: 415 593-8200
Fax: 415 553-8896
E-mail: info@aligo.com

Test Environment
Dell 1400 PowerEdge 1 CPU (795MHz), Windows 2000 Server SP1 256MB RAM

Specifications
Platforms: Windows NT/2000, various Unix
Pricing: Contact Aligo for pricing

As an industry we used to consider mobile/wireless computing to be the next frontier. Now it has become part and parcel of critical, enterprise applications ranging from Customer Resource Management to Enterprise Resource Planning. Many of the early mobile computing efforts that I've run across were built around external application service providers (ASPs) and relied on gateway technology. From an investment perspective it made perfect sense to approach the problem in this manner.

First-stage projects in new markets (such as mobile computing) often rely on "one-off" technology stacks. However, we've reached the stage where mobile computing projects should be integrated into the organization's core infrastructure. Aligo has taken this integrated approach with its M-1 Server. I recently had the opportunity to sit down with the team at Aligo and get my hands on the latest version of their software.

Aligo's Vision for Mobile Computing

Aligo's core proposition is that mobile applications must be tightly integrated with the organization's preexisting enterprise infrastructure and allow an enterprise to leverage its investment in Java resources, including developers, software, tools, and methodology. The idea is to avoid creating new mobile technology "islands" that are isolated from the mainstream of development. The Aligo approach is built around 100% J2EE technology; their product, the M-1 Server, adds mobile/wireless capabilities to any J2EE application server. While mobile applications have some unique operating characteristics, they will almost always be integrated into an organization's enterprise applications. For example, if you want to build a mobile PDA-based time card application for your mobile field force, most likely you'll integrate the time card data back into your current CRM or HR application. Adding a wireless layer to your enterprise applications presents some unique challenges:

- Mobile devices use different mark-up languages and provide a different array of operating capabilities (screen size, memory, etc.).
- The wireless network may require intermittent connectivity and often has a higher latency and lower bandwidth.
- Wireless applications offer new messaging channels (such as SMS).
- Transaction integrity into back-end enterprise databases and applications can be more complex (due to broken connections from the wireless device).

Aligo's M-1 Server addresses many of these challenges by providing a Java-based solution that can be deployed against your favorite application server, such as BEA's WebLogic, Oracle9*i*AS, and IBM's WebSphere. It also simplifies development by allowing applications written once in Java to be accessible on multiple types of mobile devices without requiring a Java Virtual Machine (JVM). The M-1 Server offers a suite of components that addresses the unique requirements of mobile applications.

The Presentation Manager layer handles the presentation across a variety of mobile devices, while the Cache Manager automatically manages the handling of dynamic content. Aligo's Device Profile Manager allows developers to provide customized profiles for each and every mobile device (i-mode, WAP phones, Palm Pilots, PocketPCs, Voice, etc.). Transaction integrity is handled by the Session Manager, which transparently manages sessions without the need for cookies. (Devices can seamlessly reconnect to applications without data loss.)

M-1 can connect to your enterprise databases through your application server – or directly to the database via M-1's Data Access Manager, which provides connection pooling, fallback recovery, and transaction rollback for JDBC-enabled data sources. One unique requirement of mobile applications is the

need for unified messaging. Aligo's Notification Manager can send unified messages through the corporate firewall to multiple message channels including fax, SMS, and voice mail.

Working with Aligo M-1 Server

The latest version of the M-1 Server is available for download and evaluation on the Aligo Web site, free of charge. You must register with Aligo before you can install the software, but it's a painless process. The installation kit is packaged as a 23MB InstallAnywhere application. The M-1 Server is built using 100% Java, so theoretically it can be deployed with your favorite application server. However, to make the software trial zip along more smoothly, the Aligo team provides a preconfigured version of the Tomcat server with the installation kit. I was able to get the M-1 Server installed and running on my Dell server in a matter of minutes.

The installation kit includes an extensive developer reference guide, but I suggest you start out with the "Getting Started with Mobile Applications" guide and the sample "restaurant" application. I think you'll find the restaurant application overview to be valuable, even if you are an experienced Java developer. Mobile applications require some different design disciplines and the sample application will help you get a feel for these differences. To access the sample application (and build your own apps) Aligo provides a specialized Application Builder for the M-1 Server as shown in Figure 1.

You can use any J2EE development tool (e.g., Sun Forte, Borland J-Builder) to build your applications. To simplify the development of mobile applications, however, the M-1 Application Builder also provides a specialized graphical user interface. To use the sample restaurant application you'll have to build and deploy it on your local development machine. As you can see from Figure 1, the application automatically launches the YoSpace Nokia phone emulator. You can also download a variety of emulators (e.g., from Openwave or Palm) to test your applications. The restaurant application prompts you to enter your name and then choose from a series of cuisines and restaurants. After you've chosen a particular place to dine, it can send you directions via fax, SMS, or e-mail.

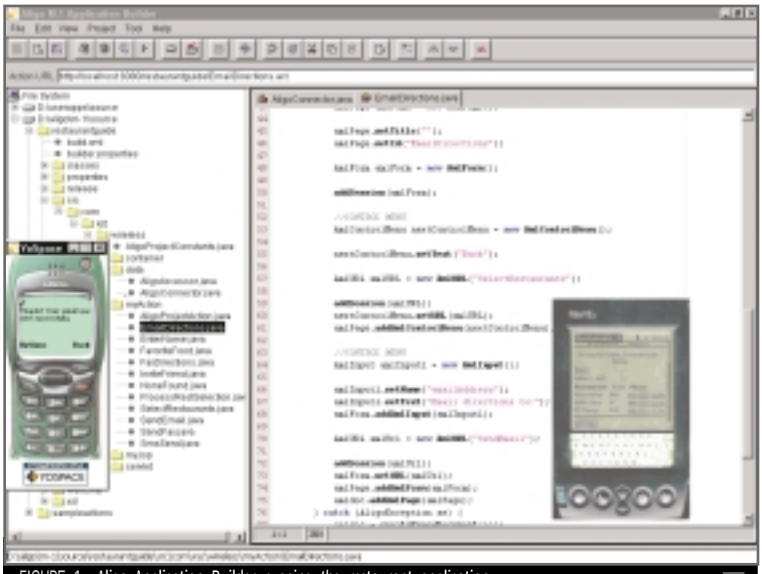


FIGURE 1 Aligo Application Builder running the restaurant application

Once I had my hands around the emulator application, I powered up my Palm VIIx and used the DPWeb browser to access the same demo application (which you can see in the overlay in Figure 1). I had the M-1 Server send me directions via e-mail to my desktop.

Aligo is completely Java-based, so Java developers should have no trouble getting acclimated to the M-1 Server platform. You're free to use M-1 with your favorite application server and to use your favorite Java IDE to develop applications. All you need to do is make the class files available to your IDE and away you go. When I visited the Aligo team in San Francisco several weeks back, they demonstrated their updated Application Builder (named M-1 Studio), which makes use of a slick drag-and-drop interface. Even if you're committed to using your existing Java IDE, I would consider trying the updated Application Builder when it becomes available (see Figure 2).

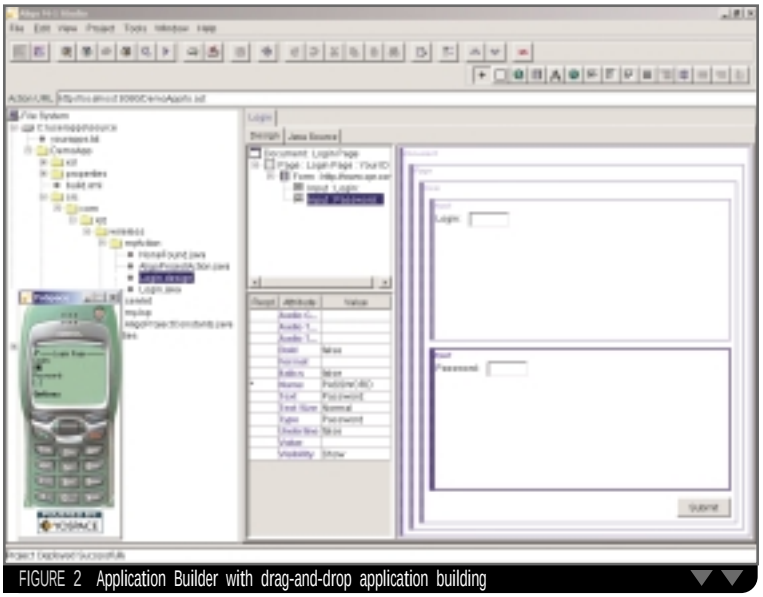


FIGURE 2 Application Builder with drag-and-drop application building

Summary

Mobile applications are here to stay and Aligo offers a compelling solution for developing wireless applications with its M-1 Server. Aligo's clean implementation using Java is compatible with your existing investments in application server technology and enterprise systems. If you're looking to add wireless computing to your systems, consider putting Aligo on your short list.

JDJ Product Snapshot

Target Audience: Java developers, mobile application developers

Level: Mid-level to advanced

Pros: Multidevice support, integration with leading application servers, mobile developer IDE

Cons: None significant

Strutting Your Stuff

Create sophisticated applications

Written by Neal Ford

In the beginning there were servlets, and it was good. They were much better than the alternatives, and allowed for scalable, robust Web development. But there was trouble in paradise.

Web development partitioned itself into two camps: *art school dropouts* (invariably Macintosh users) who could create the beautiful look and feel for the Web application, and the *Java developers* who made it work. The guys in the basement handcrafted the beautiful HTML and passed it to the developers who had to incorporate it into the dynamic content of the Web site. For the developers, it was a thankless, tedious job, inserting all that beautiful HTML into the Java code. But you drank lots of coffee and lived through it.

Then the unthinkable happened: the CEO got an AOL disk in the mail and visited a Web site he'd never been to before. Come Monday, the commandment came down from on high: we're completely changing the look and feel of the Web site. The art school dropouts fired up their Macs and started realizing the CEO's vision, and the developers got sinking feelings in the pit of their stomachs. Time to do it all over again. The problem? Too much HTML in the Java code.

JavaServer Pages

Then JSPs appeared. Here was the answer to all our prayers. JSPs have the same advantages of servlets (they are, after all, a type of servlet) and were much better at handling the user interface part of Web design. In fact, the art school dropouts could craft the HTML, save it as JSP, and pass it right to the developers. Unfortunately, all was still not well. Now the developers must deal much more directly with the display characteristics of the application. Thus the syntax of the JSP quickly becomes very cryptic with the HTML and Java code interspersed together. The verdict: too much Java in the HTML.

Model-View-Controller Pattern

Then came the Model-View-Controller design pattern for the Web. If you've been living in a cave and aren't familiar with this most famous of design patterns, here's the capsulated version: the model represents the business logic and data in the application and resides in JavaBeans and/or EJBs. The view is represented primarily by JSP pages, which have as little Java code in them as possible. In fact, all Java code should really be handled by method calls on the model beans or custom tags. The controller is the way the view interacts with the model. In the Web world a servlet is the controller.

The typical scenario for the Web MVC is: the user accesses a controller servlet, and the servlet instantiates beans, calls methods on them to perform work, adds the beans with displayable information to one of the collections (for example, the request collection), and forwards the beans to a JSP that shows the user the results.

And it was good. Now, the display information is cleanly partitioned away from the "real" work of the application, which can be strictly in JavaBeans. The application could also start using regular JavaBeans, then scale up to use EJBs without having to change the controller or presentation layers. This is clearly the best way to build Web applications. It's easy to maintain and update, and there's very little impact when one part of the system needs to change (now the art school dropouts have to worry about the new look and feel, not the developers). This design pattern neatly modularizes the constituent parts of Web applications.

Problems in Paradise

Now what's wrong? The problem with the MVC Web applications (frequently called "Model2," to distinguish it from MVC

> "Unless you're clever about it, your 'uber servlet' becomes a massive set of 'if…else' statements"

for regular applications) has to do with how you architect the Web application. For example, if you create a different controller servlet for each page the user wants to visit, you end up with dozens or hundreds of servlets that look almost identical.

Another problem is that these servlets, once visited, permanently reside as objects in the servlet engine. An alternative is to create one monster controller servlet to handle all the requests. The problem here is that you have to figure out a way to map the requests to different views. This is frequently done with parameters sent to the Web site, identifying what command you want to execute. But, unless you're clever about it, your "uber servlet" becomes a massive set of "if…else" statements or a huge "switch…case" statement. Any changes require editing this servlet, which quickly becomes unruly and ugly. What's needed is an application framework for Web development that handles most of these gory details. And that's where Struts comes in.

Struts

Struts is a Web application framework created as part of the Jakarta project at Apache. It handles 35–40% of the plumbing code required to create clean Model2 Web applications. This article provides an overview of the main parts of Struts. Because Struts is an open source project, you can download it from http://jakarta.apache.org/struts/ and start using it. For this article I cover the Struts setup, the controller servlet, form handling, custom tags, and internationalization, which only scrapes the top of the Struts iceberg.

Struts Setup

To set up Struts, you must download it and place the struts.jar file in a location where it can be loaded by the servlet engine (in other words, it must be on the classpath for the servlet engine). As an alternative you can place the JAR file in the Web application's lib directory, where the servlet engine will automatically load it. Next, if you plan to use any of the Struts custom tags, the .TLD files for the libraries you're using should be placed in your application's WEB-INF directory (more about what the custom tag libraries do for you later).

Next, you need to make some modifications to the WEB.XML file for your application. The changes required are shown in Listing 1. The first entry registers the Struts ActionServlet, which is the single point of entry for the application. Basically the Struts developers have already written the only controller you'll need. More about exactly what it does in a moment.

Controller Servlet

Several initialization parameters are passed to the controller servlet. The first specifies where to find the struts-config.xml document. This is the mapping document used by the controller. The second parameter specifies the debug level – the higher the number, the more debugging information is sent to the servlet engine log file. The third parameter is the mapping value. This points to the class responsible for handling RequestActionMapping (more about this later as well). Last, the load-on-startup tag ensures that the servlet engine loads the Struts controller as it starts up. This saves time on the first invocation of the servlet by preloading it in memory. The number passed here isn't particularly important (unless you have some dependent startup classes that depend on the controller being in memory – if so, make sure the controller servlet has a lower number).

The next entry in the WEB.XML file specifies how to let the servlet engine know that a Struts resource has been requested (as opposed to some other content from the Web site). There are two ways to handle this: prefix mapping or extension mapping. With *prefix mapping*, every resource that appears after a certain prefix will be sent to the Struts controller. This is the way that servlet engines specify that a servlet has been requested (the servlet resource always appears after a "/servlet" prefix).

Extension mapping (which we're using here) maps all resources with a particular extension to the Struts controller. This works fine as long as you pick an extension that doesn't interfere with any other registered file type. In this case we're mapping all requests that end with ".do" to the Struts controller. The ".do" extension is arbitrary (it implies that a resource will "do" something), but it's the one used by all the Struts examples so it's a bit of a Struts standard.

The remaining additions to the WEB.XML file specify the locations of the .TLD files for the custom Struts tags. Obviously you have to list only the tag libraries you're using for this application (although it doesn't hurt to list all six).

The next configuration item needed for the Web application is the struts-config.xml document. A sample version of this document is shown in Listing 2. It contains all the mappings for the controller servlet. In other words, these are the names of the resources you can request from the Web application. To understand these mappings, a discussion of Struts actions is in order.

A classic problem in software engineering is how to handle a large number of mutually exclusive choices gracefully. This same problem popped up earlier when we were talking about how to create a single point-of-entry controller. The naive approach creates a huge switch statement or a long cascading series of if…else statements. However, a design pattern called *Command* solves this problem nicely.

The Command design pattern uses inheritance instead of decisions to handle the possibilities. Typically, the developer creates an abstract class with a method called "execute()", which has no code. Then, all the different possible actions subclass the abstract class and add concrete code to the execute() method. The mapping then becomes the name of the class to instantiate. This pattern makes it easy to keep a list of all the commands (because they all have a common ancestor) and iterate through it. This is the pattern used by Struts to handle the request mappings.

In Struts you never need to create a controller servlet – the Struts developers have already created it for you. Instead, you create subclasses of the Struts Action class (org.apache.struts.action.Action). These action subclasses map to a particular resource, and this is one of the items in the struts-config document.

When a request comes into the controller servlet, it matches the name of the request (notice that the mapping name in the

"So, are we in Nirvana yet? Not completely, but we're getting closer all the time"

config file doesn't include the struts-specific extension, in our case ".do") and instantiates an instance of the appropriate action class. This means that the controller servlet never has to change to add new actions to the Web application. All that's required is the new action classes and an updated config file. Your action class takes on the role of the controller in the traditional Model2 architecture. In other words, it creates the bean (or beans) necessary to do work, calls methods on them, adds them to a collection, then forwards them to the appropriate display resource (typically a JSP). Listing 3 shows a typical small Action subclass.

As shown in Listing 3, the first step is to create a model bean. In this application the ScheduleBean encapsulates a connection to the database to get scheduling information. The Action creates an instance of the schedule bean, populates it, and passes it to the display JSP through the ActionForward class in Struts. This class is a convenience class that makes it easy to forward to other resources. Notice in the config document that the "sched" action is mapped to this action class. Thus, when the user requests "sched.do", the controller servlet instantiates the ViewScheduleAction class and calls its perform() method, which does the actual work.

The JSP is shown in Listing 4. It pulls the scheduleBean from the request collection (where the Action class placed it) and also creates another object of page scope. The main thrust of this page is to show the schedule items from the database in an HTML table. To that end the scheduleBean has a method that returns a java.util.List of ScheduleItem objects, which encapsulate the details of a single record. It would be easy enough here to create scriptlet code in the JSP to iterate over that collection. However, one of the stated goals of using this architecture is to remove Java code from the display. This is easily facilitated via one of the custom tags created by Struts.

The ScheduleView JSP uses the Struts iterate tag, a powerful custom tag that can iterate over any array or collection. The "id" specifies what the local name of the object pulled from the collection will be. The "type" specifies what class the object pulled from the collection will be cast as, and the "collection" is a JSP expression specifying the collection itself. Within the body of the tag you can use the "id" field to access any of the items from the collection without worrying about type casting – the tag has already done that for you.

To summarize, the iterate tag iterates over the collection, pulling each object out one at a time, casting it to the type specified, and assigning it to the variable name specified in "id". The user can use this variable to call any of the methods of the objects in the collection. Obviously, this is meant to be used on homogeneous collections (or at least collections that contain objects with a common super class).

This is the basic operation of a simple case in Struts. However, Struts provides even more services for handling forms, validation, and internationalization.

Handling Forms

One of the common chores in Web applications is the handling of forms. The JSP mechanism of automatically populating the fields of a JavaBean with form post parameters goes a long way in reducing the tedious code that would otherwise have to be written. Struts has built on top of this mechanism to provide an easy way to handle populating beans in wizard-style interfaces and handling validation.

When creating a Struts form you have the option of creating a Struts form bean. This is a regular JavaBean with typical accessors and mutators for the fields of the bean (and very little other code) that subclasses from the org.apache-.struts.action.ActionForm base class. Note that these beans should not connect to a database to populate themselves or provide a lot of other infrastructure code. They should be lightweight classes that basically encapsulate entity information. Other beans should handle persistence and other duties. In fact, it's typical to have one bean that acts as a collection of the entities represented by a Struts form bean (like the ScheduleBean used in the Action class discussed earlier).

One of the additional sections in the struts-config XML document allows you to declare a class as a form bean. Look at the top of the sample struts-config document in Listing 2. A form-bean declaration allows you to specify a name for the form bean and map that to a specific class (in this case, schedule.ScheduleItem). In the action mapping section of the config file you can automatically associate a form bean with an action. Notice the "add" action in the config file. The additional "name" parameter allows you to specify a form bean to use with that action. Once you have a form bean associated with an action, Struts will perform the following services for you before invoking the action method:

- Check the user's session for an instance of the bean under the name specified in the struts-config file. If one doesn't exist yet, Struts creates one and adds it to the user's session.
- For every request parameter that matches one of the setXXX methods of the bean, the appropriate set method will be called.
- The updated ActionForm bean is passed to the Action as a parameter.

Custom Tags

These services are similar to the standard JSP behavior of handling request parameters that map to JavaBean fields. However, Struts performs more services for you, and it also comes with a collection of custom JSP tags split into four categories. One of the categories allows you to replace standard HTML tags for input elements with "smarter" Struts tags. If

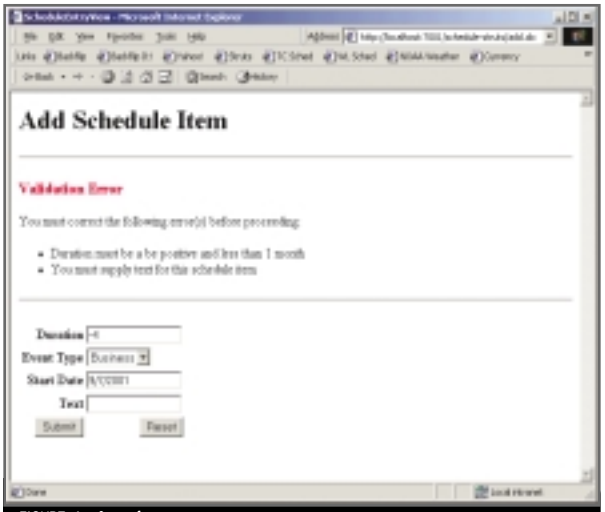


FIGURE 1 Strut form

> "It's easier than ever to create sophisticated applications in which the presentation layer is nicely partitioned from the working part of the application"

you use the Struts HTML tags, they'll also automatically populate the input fields on the form from the ActionForm whenever the page is visited. This makes it easy to handle wizard-style interfaces.

Notice that an ActionForm bean doesn't have to correspond to a single page. More typically, it corresponds to a single set of user information, so you can have an ActionForm that spans multiple pages. Using the Struts HTML tags, the input fields the user has already filled in will be automatically populated as the user moves back and forth between the pages of the wizard. Listing 5 provides an example of a JSP that uses the custom tags.

Listing 5 shows some other features of Struts tags as well. One of the automatic features of Struts is form validation. The struts-config file allows you to flag an action associated with a form bean to enable validations. This assumes that you've added a validation method to your form bean class. A sample validation method is shown in Listing 6. The validate() method is automatically called by the Action object after the population of the form bean fields, but before any of the code in the Action is performed. If the validate() method returns either null or an empty ActionErrors collection, processing of the Action continues normally. If errors have been returned from validate(), Struts will automatically return the user to the input form, repopulate the fields from the form bean, and print out a list of the reported errors at the top of the page (see Figure 1).

Figure 1 is a Struts form in which the user has put in a negative duration and left the text field blank. The errors listed at the top are automatically generated via the validate() method and the <html:errors/> tag at the top of the file (the position of this tag determines where the errors will appear on the page).

You'll also notice that the error messages returned by the validate method aren't the messages that appear on the page. The strings added in the ActionError constructor map to messages in a java.util.Properties file. This properties file is automatically referenced by Struts (this is one of the parameters of the ActionServlet), and allows for easy separation of the messages from the code in the application. This means that the text of the message can be easily changed without recompiling the application. This is also how Struts handles internationalization.

Struts can be set up to look for a properties file that matches the locale encoding of the request to automatically provide text messages in the appropriate language. This is another service provided by the HTML tags. Notice in Listing 5 that the input text references fields in the properties file. Struts includes custom tags for the four categories shown in Table 1.

Custom tags make it much easier to use Struts. In the examples above, I referenced some of the logic and HTML tags. However, because these are Struts extensions, most HTML editors and development tools don't directly support them. This is changing. Dreamweaver now has support for custom tags in general and Struts custom HTML tags specifically. You can download a plug-in that allows Dreamweaver to directly manipulate HTML elements as Struts HTML tags. This allows your HTML developers (whether they're Macintosh users or not) to easily support this application framework.

Conclusion

So, are we in Nirvana yet? Not completely, but we're getting closer all the time. Struts isn't a huge, restrictive framework. I'd estimate that it handles about 35–40% of the common drudgework of creating a Web application. On the plus side, because it doesn't try to do absolutely everything in one framework, it's very flexible. Now that tools are starting to support it, it's easier than ever to create sophisticated applications in which the presentation layer is nicely partitioned from the working part of the application.

Struts is nice because it enforces the Model2 architecture by making it hard not to follow the correct design. Applications written this way take a little longer to create up front, but the maintenance is a comparative breeze. Once developers get used to designing Web applications correctly and see the return on investment, they're happy to do things the right way.

Author Bio

Neal Ford is the vice president of technology at the DSW Group. He's also a designer and developer of applications, instructional materials, magazine articles, video presentations, and the author of Developing with Delphi: Object-Oriented Techniques *and* JBuilder 3 *Unleashed.*

nford@thedswgroup.com

TAG LIBRARY	DESCRIPTION
Bean	This tag library contains tags useful in accessing beans and their properties, as well as defining new beans (based on these accesses) that are accessible to the remainder of the page via scripting variables and page scope attributes. Convenient mechanisms to create new beans based on the value of request cookies, headers, and parameters are also provided.
HTML	This taglib contains tags used to create struts input forms, as well as other tags generally useful in the creation of HTML-based user interfaces.
Logic	This tag library contains tags that are useful in managing the conditional generation of output text, looping over object collections for repetitive generation of output text, and application flow management.
Template	This taglib makes it easy to compose pages based on templates. This is handy for pages that need common headers, footers, and side bars.

TABLE 1 Custom Tags

Listing 1: The additions to WEB.XML required to use Struts

```
    <servlet>
      <servlet-name>action</servlet-name>
      <servlet-class>
        org.apache.struts.action.ActionServlet
      </servlet-class>
      <init-param>
        <param-name>config</param-name>
        <param-value>
          /WEB-INF/struts-config.xml
        </param-value>
      </init-param>
      <init-param>
        <param-name>debug</param-name>
        <param-value>2</param-value>
      </init-param>
      <init-param>
        <param-name>mapping</param-name>
        <param-value>
      org.apache.struts.action.RequestActionMapping
        </param-value>
      </init-param>
      <load-on-startup>2</load-on-startup>
    </servlet>



    <servlet-mapping>
     <servlet-name>action</servlet-name>
     <url-pattern>*.do</url-pattern>
    </servlet-mapping>

    <taglib>
      <taglib-uri>
        /WEB-INF/struts-bean.tld
      </taglib-uri>
      <taglib-location>
        /WEB-INF/struts-bean.tld
      </taglib-location>
    </taglib>
    <taglib>
      <taglib-uri>
        /WEB-INF/struts-html.tld
      </taglib-uri>
      <taglib-location>
        /WEB-INF/struts-html.tld
      </taglib-location>
    </taglib>
    <taglib>
      <taglib-uri>
        /WEB-INF/struts-logic.tld
      </taglib-uri>
      <taglib-location>
        /WEB-INF/struts-logic.tld
      </taglib-location>
    </taglib>
    <taglib>
      <taglib-uri>
        /WEB-INF/struts-template.tld
      </taglib-uri>
      <taglib-location>
        /WEB-INF/struts-template.tld
      </taglib-location>
    </taglib>
```

Listing 2: A simple STRUTS-CONFIG.XML document

```
<struts-config>
    <form-beans>
        <form-bean
            name="addItem"
            type="schedule.ScheduleItem" />
    </form-beans>

    <action-mappings>
        <action
            path="/sched"
            type="schedule.ViewScheduleAction" />
        <action
            path="/schedEntry"
            type="schedule.ScheduleEntryAction" />
        <action
            path="/add"
            type="schedule.AddToScheduleAction"
            name="addItem"
            input="/ScheduleEntryView.jsp"
            validate="true" />
    </action-mappings>

</struts-config>
```

Listing 3: A small Action subclass

```
package schedule;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.*;

public class ViewScheduleAction extends Action {

  public ViewScheduleAction() {

  }

  public ActionForward perform(
   ActionMapping mapping, ActionForm form,
   HttpServletRequest request,
   HttpServletResponse response)
    throws java.io.IOException,
    javax.servlet.ServletException {
    ScheduleBean sb = new ScheduleBean();
    sb.populate();
    request.setAttribute("scheduleBean", sb);
    return new ActionForward("/ScheduleView.jsp");
  }
}
```

Listing 4: The View JSP for the schedule application

```
<HTML>
<HEAD>
<%@ taglib uri="/WEB-INF/struts-logic.tld"
    prefix="logic" %>

<jsp:useBean id="scheduleBean" scope="request"
             class="schedule.ScheduleBean" />
<jsp:useBean id="scheduleItem" scope="page"
             class="schedule.ScheduleItem" />
<TITLE>
Schedule View
</TITLE>
</HEAD>
<BODY>
<H1>
Schedule
</H1>

<table border="2">
    <tr bgcolor="green">
        <th>Start Date</th>
        <th>Duration</th>
        <th>Text</th>
        <th>Event Type</th>
    </tr>

<logic:iterate id="schedItem"
       type="schedule.ScheduleItem"
       collection="<%= scheduleBean.getList() %>">
        <tr>
          <td><%= schedItem.getStart() %></td>
          <td><%= schedItem.getDuration() %></td>
          <td><%= schedItem.getText() %></td>
          <td><%= schedItem.getEventType() %></td>
        </tr>
</logic:iterate>
</table>
<p>

<a href="schedEntry.do"> Add New Schedule Item</a>

</BODY>
</HTML>
```

Listing 5: An HTML form using the Struts HTML custom tags.

```
<%@ taglib uri="/WEB-INF/struts-html.tld"
    prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"
    prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts.tld"
    prefix="struts" %>
<jsp:useBean id="scheduleItem" scope="request"
             class="schedule.ScheduleItem" />
<jsp:useBean id="scheduleBean" scope="page"
             class="schedule.ScheduleBean" />
<% pageContext.setAttribute("eventTypes",
 scheduleBean.getEventTypes()); %>

<HTML>
<HEAD>
<TITLE>
```

```
ScheduleEntryView
</TITLE>
</HEAD>
<BODY>
<H1>
Add Schedule Item
</H1>
<hr>
<html:errors/>
<html:form action="add.do">
<table border="0" width="30%" align="left">
  <tr>
    <th align="right">
      <struts:message key="prompt.duration"/>
    </th>
    <td align="left">
      <html:text property="duration"
                        size="16"/>
    </td>
  </tr>
  <tr>
    <th align="right">
      <struts:message key="prompt.eventType"/>
    </th>
    <td align="left">
      <html:select property="eventType">
        <html:options collection="eventTypes"
                  property="value"
                  labelProperty="label"/>
      </html:select>

    </td>
  </tr>
  <tr>
    <th align="right">
      <struts:message key="prompt.start"/>
    </th>
    <td align="left">
      <html:text property="start"
                           size="16"/>
    </td>
  </tr>
  <tr>
    <th align="right">
      <struts:message key="prompt.text"/>
    </th>
    <td align="left">
      <html:text property="text"
                           size="16"/>
    </td>
  </tr>

  <tr>
    <td align="right">
      <struts:submit>
        <bean:message key="button.submit"/>
      </struts:submit>
    </td>
    <td align="right">
      <html:reset>
        <bean:message key="button.reset"/>
      </html:reset>
    </td>
  </tr>
</table>
</html:form>

</BODY>
</HTML>
```

Listing 6: The validation method for the ScheduleItem form bean

```
public ActionErrors validate(
         ActionMapping actionMapping,
         HttpServletRequest request) {
    ActionErrors ae = new ActionErrors();
    if (duration < 0 || duration > 31) {
         ae.add("duration", new ActionError(
            "error.invalid.duration", "8"));
    }
    if (text == null || text.length() < 1) {
         ae.add("event text",
              new ActionError("error.no.text"));
    }
    return ae;
}
```

Download the Code! www.JavaDevelopersJournal.com

JClass ServerChart 1.1

by Sitraka Software

REVIEWED BY JIM MILBERY
jmilbery@kuromaku.com

It's often said that "a picture is worth a thousand words" – and I've generally found this to be true. In fact, it's often easier for users to digest numeric data when it's displayed in picture form, such as with a chart or graph. Business analysts have long understood this concept and often choose to display raw data in graphic form to make it easier to understand. With the popularity of Java servlets and JavaServer Pages, it's now possible to dynamically display this very same data on the Web. Sitraka Software is one of the premier software vendors in the Java components market. They market a powerful dynamic-charting component called JClass ServerChart, which is perfect for generating colorful charts and graphs for the Web.

JClass ServerChart

JClass ServerChart is a charting and graphing component that's written entirely in Java. Developers can use the ServerChart Beans within popular IDEs to enhance Java client applications (or applets). They can also use these same beans within servlets and JSP to build highly visual, dynamic Web page applications featuring charts and graphs. ServerChart allows you to set chart properties programmatically, and also instantiate and extend the JClass ServerChart objects within code. The software supports all the popular chart formats including Plot, Scatter Plot, Area, Stacking Area, Bar, Stacking Bar, Pie, Hi-Lo, Hi-Lo-Open-Close, and Candle. You can also create custom charts by manipulating one of these standard formats.

Installing and Using ServerChart

Sitraka makes the JClass ServerChart software available for download from the Sitraka Web site, but I used an installation CD to install and configure it. Although you can use the ServerChart JavaBeans within a Java application on the client layer, the software really shines on the server side. Thus I elected to configure ServerChart to work with an application server on my laptop. It's not difficult to configure any of the popular servlet engines to work with ServerChart. (In fact, JClass even supports the database connection pooling capabilities of IBM WebSphere and BEA WebLogic). However, Sitraka makes it even easier to get started by including a preconfigured version of Tomcat with the ServerChart installation kit. I elected to use the built-in Tomcat server, which made the installation a breeze. It took me about five minutes to get ServerChart running (and this included three minutes trying to get the shrink-wrap off the box). Sitraka continues to impress me with its understanding of the "out-of-the-box experience." Not only is the software easy to use, it's equipped with tons of examples and demonstration charts. Lots of Java software that I review is aimed squarely at the high-end developer, but JClass ServerChart is perfect for both novices and professionals. In fact, Sitraka provides a great HTML-based utility example that you can use to get rolling with ServerChart. I was able to fill in data values (and formatting information) on a simple HTML page and generate a complete bar chart as shown in Figure 1.

JClass ServerChart generates charts in a variety of graphic formats, including PNG and JPEG. Thus the browser client doesn't require any special configuration to view the generated charts and graphs. Although the sample chart that I built was very simple, it's just as easy to build very sophisticated charts with JClass ServerChart. The installation kit includes over a dozen examples of advanced charts, such as the combination line/multibar chart shown in Figure 2.

Sitraka gives you almost complete control over the formatting of the chart object. You can even create image maps that cause your charts and graphs to have powerful "click-throughs" in the display. Thus you could display a summary chart of data to a user, and then allow the user to drill down into the underlying data for each section of the graph.

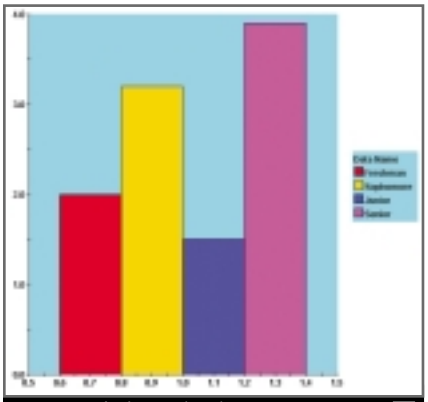


FIGURE 1 Simple NetU bar chart

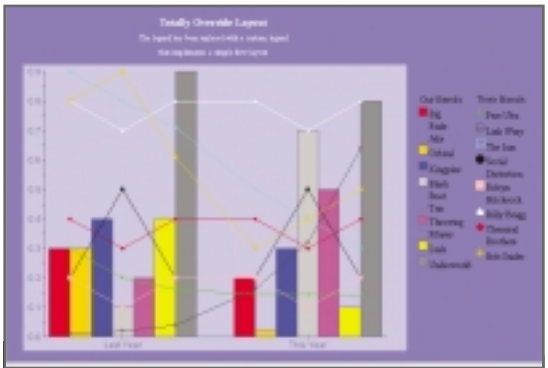


FIGURE 2 JClass ServerChart multiformat graph

specs

Sitraka Software
260 King Street East
Toronto, Ontario, Canada
M5A 4L5
Phone: 800 663-4723
Fax: 416 594-1919
Web: www.sitraka.com

Test Platform
Toshiba Satellite Pro 4600, 866MHz Intel Pentium III processor, 20GB Disk, 256MB of memory, Windows 2000 w/ Service Pack 2

Specifications
Platforms: Any platform with JDK 1.1, 1.2 support
Pricing: $699

JClass ServerChart 1.1 by Sitraka Software

JClass provides a wealth of additional features and functions that are just too numerous to list here.

Summary

Sitraka has always impressed me with its product quality – JClass ServerChart is no exception. Dynamic charts and graphs are ideal for many applications, and there's no reason to "reinvent the wheel" by building these components yourself. It's an ideal product for creating charts and graphs. If you're looking to add these features to your Web applications, I suggest you put Sitraka's JClass ServerChart on your shopping list.

JDJ Product Snapshot

Target audience: Java programmers, Webmasters, business analysts
Level: Beginner to advanced
Pros: Feature-rich product, preconfigured servlet support, powerful charting and graphing capabilities, tons of useful examples and demonstrations
Cons: Nothing significant

The Many Sides of J2EE Development

Bringing great new insights and possibilities to the table

WRITTEN BY
CHARLES AREHART

There are many facets to J2EE Web application development. It's a powerful platform that offers a variety of possibilities and capabilities, with many different approaches and models of development available. This is both a strength and, for newcomers to the platform, an Achilles' heel.

In this installment of **Journeyman J2EE**, we look at the many sides of J2EE development with a focus on the diversity, dichotomy, and divided opinions that can challenge even experienced developers, but especially newcomers. Consider these some warning signs on the road to J2EE.

So Much to Learn, Who to Believe?

In the previous **Journeyman J2EE** (***JDJ***, Vol. 6, issue 9) I discussed the importance of continuous learning as a J2EE Web application developer, especially for those making the transition from other platforms such as ASP, PHP, and ColdFusion. This extends beyond the obvious matters of learning the Java language, JSP programming, and/or the servlet API.

Of course, there are all the other APIs available in the platform, including JDBC, JMS, JNDI, JTA, and RMI, that you may or may not need to learn about at the outset. If you need to use any of them, it will be rather obvious but they're worth persuing as time permits.

You may also need to learn about EJBs, but this can be yet another source of confusion as there are many opinions about their role. More about that later.

Even the simple choice of whether to use servlets or JSPs is beset by many arguments for and against. Just when you think you may have a handle on the debates and best practices, you may encounter discussions of design patterns, which can be a great resource for applying reusable design techniques but is yet another set of practices to learn and apply. (See the J2EE Blueprints at http://java.sun.com/j2ee/blueprints/index.html for more information.)

It's enough to make your head spin if you're coming from one of the aforementioned Web development platforms. In those communities we knew the enemy and the enemy was "them" – all the developers using other platforms. Okay, that's stretching it, but as much as there may have been debate within each community about best practices for implementing apps, nothing comes close to the incredible variety of approaches (and opinions) about how best to do J2EE Web application development.

There's no use crying about it. The Java platform is incredibly rich, and this variety speaks to the many different ways in which the tools can solve problems for a wide range of applications. One of the key problems I find in discussions about alternative approaches (or rather a problem I have with naming "best practices") is that what works for one situation may not be the best solution for another.

A frequent source of confusion is the matter of whether you're programming "in the small" or "in the large." In other words, are you developing a rather simple application, displaying just a few pages to be used by a relatively small audience, or are you developing a full-blown e-commerce application with authentication and authorization, database transaction processing, session handling, personalization, internationalization, and more that will be implemented on a cluster of servers with distributed processing?

What makes sense in the former case may fall woefully on its face in the other. By the same token, the approaches taken in the latter case may be overkill for the first. It's a delicate balancing act and the decision often falls on folks who don't have the time or experience to entirely grasp the ramifications of the many approaches. It's daunting, and some are forced to make quick (and perhaps ill-informed) decisions.

Let's look a little more closely at some of the debates. We don't have time to resolve or even carefully cover all of them, nor all facets of each, but they'll give you a taste of what you're in for if you're just getting into this brave new world.

Servlets or JSPs? You Decide

As I mentioned before, the decision of how best to use servlets and JSPs is no simple task. Should you use one or the other? Or both? Many applaud JSPs for making Java server Web development easier for many people. They're similar to other Web application scripting tools and hide some of the challenges of learning Java when getting started. Indeed, others will laud that they make it possible to split development roles among Java developers and pure page designers (who should just ignore the occasional directive or action element and might even be able to create JSPs with WYSIWYG design tools).

The reality for many developers, however, is that they are the designer, Web developer, and Java programmer (or newcomer) and that separation of roles isn't as important. It's for this audience though that others will argue that JSPs will cause a developer to shoot him- or herself in the foot, littering pages with embedded Java scriptlets, JDBC calls, and otherwise corrupting the vaunted Model/View/Controller (MVC) paradigm, which calls for the separation of data display, request processing, and business logic.

Some go further and plain decry JSPs in general as a poorly designed development approach, as argued in the article "JSP: You Make the Decision," by Jon Stevens (***JDJ***, Vol. 6, issue 7), which supports the Velocity Template Language as an alternative.

Others will argue that servlets are the

only way to go; yet even among servlet fans there are debates about how to implement them. Should you have a central controller servlet through which all requests pass or not? The same argument can be debated in JSP development as well.

It seems that for most, a mix of servlets and JSPs is the way to go, and the O'Reilly title, *JavaServer Pages* by Hans Bergsten, does a great job of discussing the many different approaches (including integrating servlets with JSPs), with an especially lucid perspective that suits relative newcomers. I'd like to publicly applaud his book for doing a great service for those facing these frequently fractious debates. Notice that I'm not taking sides: my point it that you need to carefully weigh the alternatives but be sure to consider several arguments.

What About MVC?

A few more words about the MVC paradigm may be in order. If you haven't heard of it before, don't worry. You won't get far in your exploration of J2EE development before learning that for many it's a linchpin of interactive application development.

The only problem is that many MVC discussions will assert that a key facet is that the client "view" must be kept in sync with the back-end data source "model" driving its data display. Of course this can't work in the Web application model: there's no easy way to update a page sitting on your client's browser because data on the server has changed.

Does that mean the MVC paradigm is severely crippled for Web application development? No. It's just one part of the paradigm (and some may argue not even the most important). The MVC model and its separation of data display, request processing, and business logic makes sense as a desirable approach in all other respects.

Java Within JSPs

The subject of MVC and its separation of content and code leads to yet another area of debate, with respect to whether and how much Java you should be coding in a JSP. Again, this is an area that will be influenced by your perspective on whether nondevelopers will be editing a given JSP template (in which case a minimum of Java is important so as not to confuse them or put the code at risk by their modification).

Even then, if the JSP developer(s) is more savvy, does it still make sense to litter a JSP with embedded Java? Scriptlets are easy to use if you know Java, but there are several alternatives that will hide the code and it may be beneficial to consider them.

Custom tags (or custom action elements) are a well-defined feature in the JSP specification widely supported by JSP containers. They not only allow segregation and encapsulation of code, but also form the basis for resusable components and even greatly simplified processing within a JSP page.

For some the only use for custom tags is to leverage entire tag libraries from third parties (or Web container providers) that provide a whole range of useful services and features in your JSP, including easier database access, record processing, and mail handling.

Know that they are available as an option to ease your own development by providing another means to separate your code from your JSP. Still another way to segregate code is to use JavaBeans, Enterprise and otherwise.

What About Beans, EJB and Otherwise?

Ah yes, JavaBeans or EJBs. For the newcomer few things may be as confusing. It's pretty easy to learn how to use JSPs and even the servlet API, but if you're new to Java you'll need to do some work to wrap your head around these two topics.

JavaBeans and EJBs are not related to each other. Similar to the way JavaScript's name has confused people who assume it's a lightweight version of Java, so too do some think Enterprise JavaBeans are a heavyweight version of JavaBeans.

Some will argue that EJBs are heavyweight, though that's a matter for debate. In any case, they're not the same and they're also not mutually exclusive (you can use one or the other, or both, or neither).

In my last article I mentioned how disheartening it can be for a newcomer to J2EE who may read that JavaBeans are critical to JSP/servlet development, only to find books that describe them solely from the perspective of their value in client application development and/or IDE integration.

It's easy to miss the value of JavaBeans, which is, like so much in Java, about encapsulating data and methods into an object that can be used to describe something in your Web application. With their standardized mechanism for accessing data in the bean (getters and setters), they also support another fundamental goal – reusability – and, perhaps more important, hiding implementation behind a widely accepted interface.

Again I'll refer to the book, *JavaServer Pages*, for its excellent explanation of the value and application of JavaBeans, though some reviewers at Amazon have faulted it for relying too much on JavaBeans and not discussing EJBs that much. He does indeed recommend them as an approach that's suited to more enterprise class applications, and this exposes yet another area of debate.

What about EJBs? Should they by necessity be a core component of your Web application? Some will argue that absolutely no respectable Web application would be built without them. Others, though, will debate that and assert that their overhead is overkill. As you read about them you'll learn that there are different perspectives to bring to the table. Again, I can only recommend that you keep your mind open.

Another aspect of the discussion of EJBs may lead to real confusion for the newcomer: some resources will be describing EJBs

Author Bio

Charles Arehart is a 20-year IT veteran with experience spanning a range of technologies, including large scale database systems. For the past four years he's been an active trainer, writer, and consultant in enterprise Web application development. He contributes to several resources, provides on-site coaching and consultation, and is a frequent speaker at user groups throughout the country.

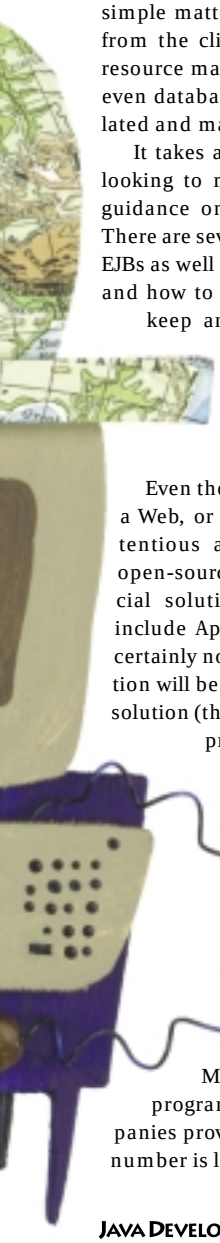
from the perspective of client application/applet developers. If these client apps need to connect to a remote distributed server, then encapsulating the business logic (and database interactions) on the server rather than the client will make eminent sense. But it's not an argument that resonates with Web application developers who have always been doing their business logic "on the server." Do EJBs then still add value?

As with the debate over MVC, purists, newcomers, and more practical adherents are often talking on different planes, making presumptions that don't apply in all cases.

EJBs make sense when both your Web and EJB containers are distributed on different systems. What about when they're on the same server? It's not that they don't make sense in that last instance, just that their benefit has to be more carefully weighed against their cost. Some containers provide integration that eliminates some of the overhead when the Web and EJB containers are on the same server.

There's clearly a lot more to EJBs than the simple matter of segregating the business logic from the client. There are aspects of security, resource management, transaction control, and even database integration that can be encapsulated and managed by EJBs.

It takes a lot of reading, trial and error, and looking to more experienced counterparts for guidance on deciding when to leverage EJBs. There are several books on the generic subject of EJBs as well as the more specific matter of when and how to use them in Web applications. Just keep an open mind that's both skeptical and flexible. However, the diversity of opinion doesn't end with the debate over EJBs.



Which Server/Container?

Even the matter of choosing a J2EE server or a Web, or EJB container is fraught with contentious and rancorous debate. There are open-source adherents for whom no commercial solution will do. Prominent offerings include Apache's Tomcat and JBossServer. It's certainly not obvious that an open-source solution will be any less capable than a commercial solution (though be careful, not all open-source products are licensed to be used for production purposes).

Even among commercial products there's a world of difference among the many contenders. Of course, the largest share of the J2EE server pie goes to BEA and IBM. There are many other worthy contenders, including offerings from Borland, HP, Iona, iPlanet, and Macromedia. Sun's J2EE-compliance program will help to sort out which companies provide fully compliant servers, but the number is large and growing.

Then, too, it's not always necessary to acquire a complete J2EE server. Perhaps your needs will be satisfied with just a servlet container. There are many offerings in that arena as well, including New Atlanta's Servlet Exec and Gefion's LiteWebServer.

Many vendors offer trial versions, and some offer unlimited development versions (or versions allowing only a small number of users). It's easy enough to try out a variety of servers to determine which suit your needs. Also, look to resources like Sun's Solutions Marketplace (http://industry.java.sun.com/solutions) and *JDJ*'s Buyer's Guide (www.sys-con.com/java/wbg/index.cfm) for more insights.

What's It All Mean?

Is all this variety and contention over different approaches a bad thing? Not really, it's just different if you've only recently come into the J2EE arena. Is it hopeless? No, there are many resources available to help you learn about the many approaches, including articles, books, Web site information portals, classes, and mailing lists. The challenge is to digest it all, keep an open mind, be open to debate, and don't give any one argument too much weight. Favor those that do a good job of offering a balance of approaches, and compare and contrast them.

If anything, these numerous possibilities are another facet of the Java platform in general. It's a rich and flexible language, a platform with much to offer, and it satisfies an incredibly wide audience. And that audience is expanding all the time.

If there's one more hopeful message to sound, it's that this very widening of the Java community will surely lead to a more cohesive and well-digested sense of how best to apply the many sides of J2EE to a range of problems.

The fact that Java is set up to be molded and influenced by the Java Community Process is a sign that hopefully more and more public discussion by an ever-widening (and constantly learning) circle of participants will yield still better implementation solutions.

Fasten your seatbelts, fill your coffee cups, and settle in for the ride, which will be anything but boring. There will be an ever-growing abundance of resources and experienced cohorts to learn from, and even the newcomers among the audience may bring great new insights and possibilities to the table.

One last thing: speaking of the variety of developers out there, we welcome your feedback, not just to this article but to all of them. You'll notice that if you visit the *JDJ* Web site (www.JavaDevelopersJournal.com) you're now able to share your comments on all the articles, and subscribers can even rate them on a scale of 1 to 3. Please take a moment to share your feedback on any articles that strike you one way or another. It's all a part of the opportunity for continuous learning.

carehart@systemanage.com

WRITTEN BY JOHN ZUKOWSKI

The Magic of Merlin

Stop the presses! Clear the rain forests! The Merlin books are coming! The Merlin books are coming! This thing is huge and the books will keep getting bigger. Forget about thin clients. The runtime class libraries alone have jumped from 13.5MB in the 1.3 release to 22MB with the new version. Looking back to the Fall of '95, the entire Java 1.0 download was less than 4MB, about one-third the size of the library additions, and the 4MB also included the compiler and source code. Alone, the version 1.0 libraries were just 1.4MB. Boy, have the Sun engineers in Santa Clara and around the globe been busy.

The documentation that comes with the beta 2 release includes a partial listing of the 400-plus bugs closed. Only those bug reports with at least one vote in the Bug Parade at the Java Developer Connection are listed. Apparently some things were important enough to fix, even without the votes, but not important enough to list. I don't get it. I hope by the time 1.4 goes final, we'll be able to see a list of all bugs closed. Please…

What I find most interesting is how long the listed bugs have remained open. With Sun's release cycle between versions now at about 18 months, anything older than that falls back on problems or enhancements requested against older versions. While the reports don't list the date fixed, you can think the bug reports aren't really closed until 1.4 becomes final, so these numbers will actually increase a little.

What is that average? When I calculated the numbers on September 24, it was 767 days, or over 8 months prior to when version 1.3 was released. If you calculated a weighted average against the number of votes, the average gets closer to 1,000, at 970 days.

For the curious, the most voted for enhancements added and bugs fixed are assertions, a fix of NT services hanging when users log out, and a speedup of remote double buffering. On the age side, the oldest enhancement request was open for 1,800 days (almost 5 years). The oldest bug was open for a little over 1,700 days.

When I previously commented about the long delay to fix bugs (and add enhancements) in an IBM developerWorks article (www106.ibm.com/developerworks/java/library/j-mer0717), a Sun engineer wrote, "Our enhancements have always been based on developer demand and community pressure." While I tend to demand a great deal, I've always thought Sun (and the JCP community) was more interested in filling out what was missing from the Java platform than in fixing up what was already there, even if what was there was incomplete. With Merlin, it seems like they're finally going back and fixing or filling in many of the little things that have been driving us nuts. For that I say thanks.

If you haven't moved to 1.3 yet, should you just skip ahead to 1.4? I'd have to say a definite yes to that. It's still too early to see what the memory footprint requirements will be, but as far as features go, the 1.4 release seems to have it all. Of course, there will be more new capabilities added for the Tiger (1.5) release, so don't think that Sun is through just yet. Let's hope they don't add too many new bugs in the process. I haven't logged nearly as many new bugs with the 1.4 release as I did with the earlier versions during their beta cycles. Let's take that as a sign that the software is improving prior to the beta release.

One final thing worth mentioning is the latest Java plug-in version that Sun is trying for the 1.3 release. An HTML converter is no longer necessary. The <APPLET> tag will work with an updated JVM within browsers. This is great news for those users confused about Java support with Windows XP and IE6.

george.grey@savaje.com

AUTHOR BIO

John Zukowski conducts strategic Java consulting with JZ Ventures, Inc. His latest books are Java Collections and Definitive Guide to Swing for Java 2 (2nd edition) from Apress, with two books due out in early 2002: Learn Java with JBuilder 6 (Apress) and Mastering Java 2, JDK 1.4 (Sybex).

J2SE INDEX

Add Fine Grained Access Control for Methods

Use the contents of a trace to find information about the caller

WRITTEN BY
ASHUTOSH NARHARI

Java designers have been carefully providing access modifiers for methods, classes, and instance and class variables. In almost all cases these modifiers are sufficient when designing object-oriented applications.

However, for methods that need specialized access control you may want something better than Java's access modifiers. In this article I demonstrate how to implement access control for your methods in a slightly nonconventional way. (The source code for this article can be found on the ***JDJ*** Web site, www.sys-con.com/java/sourcec.cfm.)

Access Modifiers in Java

Java provides four access levels via three access modifiers: public, protected, and private. When a method doesn't have an access modifier specified, it's said to have package (default) access. When applied to a method, these modifiers result in a unique type of access. Let's look at the access levels:

1. ***Public:*** A public method can be accessed by any class in any package and is the least restrictive type of access.
2. ***Private:*** A private method can only be accessed by the instances of the class that define the method; even subclasses can't access this method. This is the most restrictive type of access.
3. ***Protected:*** A method with a protected modifier can be accessed via the class implementing that method, all classes in the same package as the implementing class, and any subclass of the implementing class.
4. ***Default/Package:*** When no access modifier is specified for a method, that method is visible to all the classes in the same package as the class that implements that method.

If we want any access control beyond what access modifiers offer, we'll have to implement our own. The trick is to examine the stack trace to find out information about the caller and then approve or deny access based on that information. Let's see how we can implement that with the help of stack traces.

Stack Traces: Not Just for Debugging

Examining stack traces is a common chore for Java developers. Stack traces are a snapshot of the JVM's threads/monitors at the instant a trace is created. We've seen those traces when an exception occurs (mostly when a runtime exception occurs or when an exception is caught and the stack is dumped to the console or a log). Stack traces are great for finding out where the exception occurred and what was in the stack at that time.

The information in a stack trace helps you discover various method calls that lead to the problematic line of code causing the exception. But we don't have to wait for an exception to occur to get a stack trace. Using the following method we can get one without throwing an exception.

```
Throwable t = new Throwable();
t.printStackTrace();
```

This would print something similar to the following on the console:

```
java.lang.Throwable
        at
jdj.checkpoint.CheckPoint.<init>(Ch
eckPoint.java:34)
        at
testpackage.Test.methodB(Test.java:
24)
        at
TestSimple.main(TestSimple.java:28)
```

Contents of a Stack Trace

If you need to examine the trace programmatically, get it in a string. You can use a method similar to the following to get a string containing a stack trace from an instance of Throwable or any descendants of Throwable. (Exception is a subclass of Throwable.)

```
public String
extractTraceString(Throwable t) {
 StringWriter stringWriter = new
StringWriter();
 PrintWriter printWriter  = new
PrintWriter(stringWriter);
 t.printStackTrace(printWriter);
 return stringWriter.toString();
}
.
```

Controlling Access

If at the beginning of each method we could check for the caller's identity, we can determine whether the method should continue executing or just return with an exception.

Let's use the information we just summarized earlier and put it to use by creating a class that checks for the caller's package, class, and method.

The idea is to instantiate an object (let's call the class CheckPoint) and provide the allowable package(s), class(es), and method(s). Then a simple method call should be enough to determine if the method should continue executing or not.

Let's assume we have a class named PickyClass that contains a method pickyMethod(). We want any objects of class "FriendClass" to be able to call pickyMethod, but all other classes shouldn't be allowed to execute that method.

Once we're done with our class CheckPoint, we should be able to do something like:

```
public void pickyMethod() {
```

```
 CheckPoint checkPoint = new
CheckPoint();
checkPoint.allowClass("FriendClass"
);
 If (!checkPoint.isCallerValid()) {
//take necessary action to let the
caller know that they
//don't have access.
 }

 //rest of the pickyMethod()
}
```

We should have the methods allowPackage(String) and allowMethod (String) as well. Also, we can add methods such as checkAccessAndThrowException(), which can throw an exception so pickyMethod can change its signature to throw an exception and the callers can check for it.

We'll use hashtables to accept a list of packages/classes/methods. There are other ways to do it but I chose hashtables for two important reasons. First, when the time comes to check the caller against these lists, the hashtable can provide a fast lookup. Second, it would be nice to keep CheckPoint compatible with Java 1.1.

> The trick is to examine the stack trace to find out information about the caller and then approve or deny access based on that information

Here are three hashtables that CheckPoint has as instance variables:

```
 private Hashtable allowedPackages;
 private Hashtable allowedClasses;
 private Hashtable allowedMethods;
```

We have three methods to populate these tables:

```
 public final void
allowPackage(String packageName),
public final void allowClass(String
className), and
public final void
allowMethod(String methodName)
```

Each of these methods simply adds a key-value pair, key being the string provided and value being an empty string. Since we'll never need the value, an empty string acts as a placeholder.

Let's start with the core functionality of CheckPoint. We know that CheckPoint should:

1. Create a stack trace at instantiation.
2. Parse a stack trace string to determine the caller's package, class, and method.
3. Keep track of packages, classes, and method names that will be allowed access.
4. Compare the caller against the allowable packages/classes/methods.
5. Allow or deny access based on the above test.

We've already looked at creating a stack trace string. We just want to make sure it gets created as soon as the object is instantiated, so let's create it in the constructor and parse it. The constructor looks like:

```
public CheckPoint() {
  allowedPackages          = new
Hashtable();
  allowedClasses  = new
Hashtable();
  allowedMethods           = new
Hashtable();
  parseFailed              = false;

  Throwable throwable = new
Throwable();
  this.stackTrace =
this.extractTraceString(throwable);
  this.parse();
 }
```

Right before we finish constructing the object we also want to parse the stack. A method call to parse() parses the trace and assigns values to three instance variables of this class, namely callerPackageName, callerClassName, and callerMethodName.

Parsing the Trace

Once we get the trace in a string, we can do some simple parsing to get the information we're looking for, namely, the caller class's package and name, and the caller method's name.

To find the correct line in the stack trace we need to ignore a few lines. Since CheckPoint's constructor creates the stack trace, we know the next line in the trace would be the one that corresponds to the line that instantiated the stack trace. And the following line contains the information about the caller of the method that's trying to check the access. In the following example trace TestSimple.main() is the caller.

```
java.lang.Throwable
        at
jdj.checkpoint.CheckPoint.<init>(Ch
eckPoint.java:34)
        at
testpackage.Test.methodB(Test.java:
24)
        at
TestSimple.main(TestSimple.java:28)
```

The method parse() (see Listing 1) does just that. It skips to the line that contains caller information and then breaks apart the caller's package, class, and method. This information is then stored in three instance variables:

```
private String callerPackageName;
private String callerClassName;
private String callerMethodName;
```

Checking the Access

Once we have a list of valid callers and the current caller's information, we can simply see if the caller exists in the lists. The method isCallerValid() checks if the caller exists in the hashtables and returns a Boolean based on its discoveries.

The key here is to perform the check only if the list provided is not empty. If a list of valid callers was provided and the caller doesn't exist in the list, we know the test failed. Here's an example of how we would check if the caller's package is valid or not:

```
if (allowedPackages.size() > 0) {
 if
(!allowedPackages.containsKey(calle
rPackageName)) {
   packageCheck = false;
 }
}
```

To provide easy checking, we don't want the user of CheckPoint to see if a check was positive or negative. We can provide a simple method checkAccess() that wouldn't return anything, but would throw a runtime exception if the check fails. This method could look like:

```
public final void checkAccess() {
 if (!this.isCallerValid()) {
  throw new
RuntimeException("Method Access not
```

```
allowed");
 }
}
```

We can also provide a method that doesn't throw an unchecked exception but actually throws a checked exception. We can create a new method that's similar to the one above but throws a MethodAccessException (a simple extension of exception), which has to be caught by the users of CheckPoint. This method looks similar to:

```
public final void
checkAccessAndThrowException()
throws MethodAccessException {
 if (!this.isCallerValid()) {
  throw new
MethodAccessException("Method
Access not allowed");
 }
}
```

That concludes all the steps necessary to create a simple class that lets us provide access based on the caller.

Usage of CheckPoint

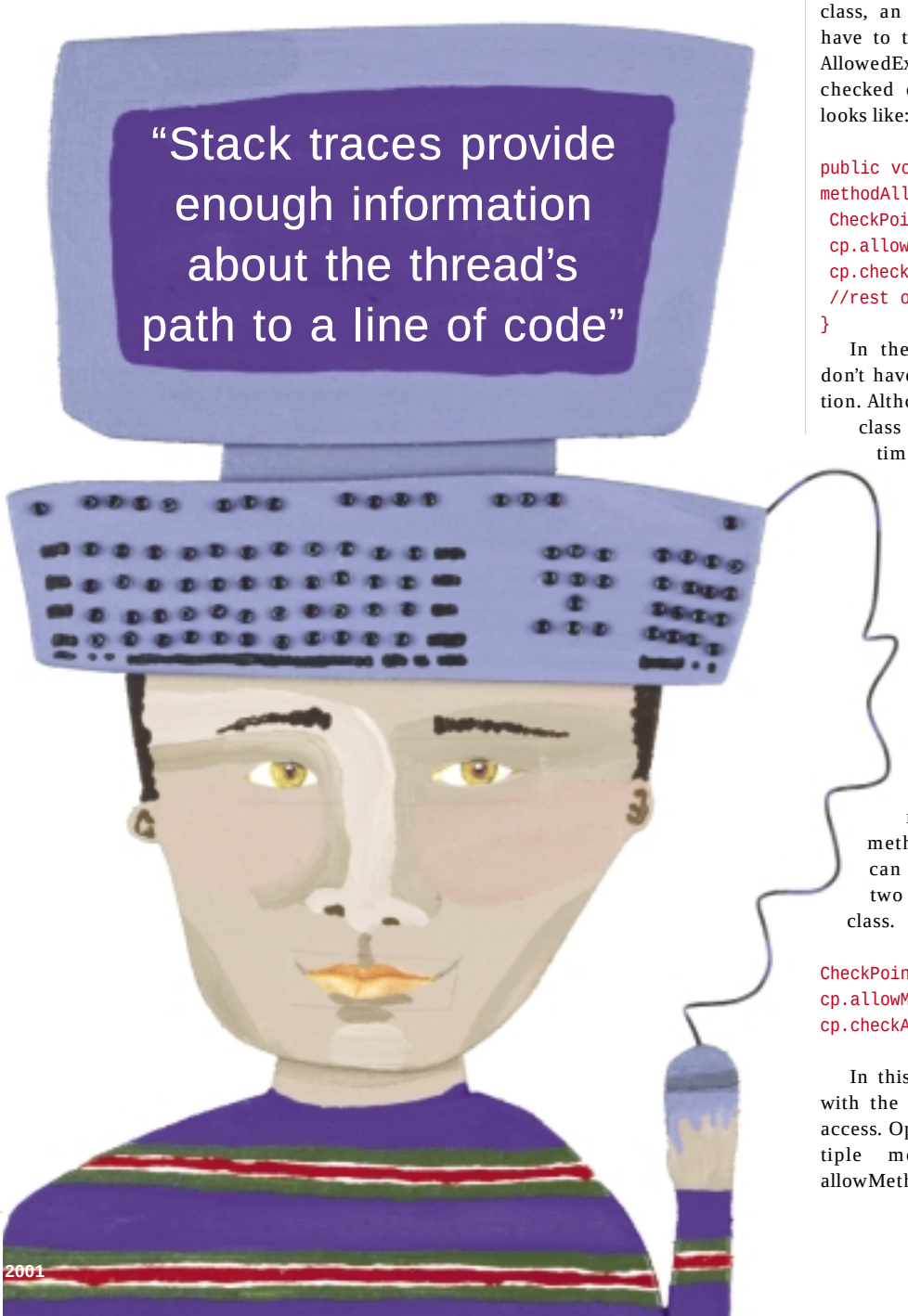
To protect a method a public modifier is specified for your method and the first thing in the implementation of the method is a check for access. If the caller is allowed access, everything after the access check is executed as normal. If access isn't allowed, an exception is thrown. As we discussed earlier, we can choose to have a checked or unchecked exception thrown. For example:

```
public void
methodAllowedFromOneClass() throws
AccessNotAllowedException {
 CheckPoint cp = new CheckPoint();
 cp.allowClass("SafeCaller");
 cp.checkAccessAndThrowException();

 //rest of the code for this method
}
```

In this example only instances of SafeCaller are allowed. If this method is called from an instance of any other class, an exception is thrown. Callers have to try and catch the AccessNotAllowedException. To generate an unchecked exception, the same method looks like:

```
public void
methodAllowedFromOneClass() {
 CheckPoint cp = new CheckPoint();
 cp.allowClass("SafeCaller");
 cp.checkAcces();
 //rest of the code for this method
}
```

In the above example, the callers don't have to try and catch any exception. Although, if the caller exists in any class other than SafeCaller, a runtime exception will be thrown.



Controlling at the Method Level, Not the Class Level

Access modifiers provide access based on where the caller class exists. This means that for all the methods of the caller class, the access to the called method is the same.

CheckPoint can provide access based on the caller method or a list of caller methods. Using CheckPoint, you can provide different access for two methods of the same caller class.

```
CheckPoint cp = new CheckPoint();
cp.allowMethod("safeMethod");
cp.checkAcces();
```

In this example, any caller method with the name safeMethod is allowed access. Optionally you can specify multiple method names by calling allowMethod() multiple times.

Controlling Access at the Class Level

Access can be controlled by checking the caller object's class.

```
public void methodAllowedFromOneClass() {
 CheckPoint cp = new CheckPoint();
 cp.allowClass("SafeCaller");
 cp.checkAcces();

 //rest of the code for this method
}
```

You can specify one or several classes that are allowed access into your method.

Controlling Access by Providing a List of Packages

Using a default/package-level access modifier for a method means that all the classes in the same package can access that method. The only way to allow access to classes in packages other than the default one is to use public or protected access. The limitation of using public is that all classes in all the packages have access to your method. Using protected allows you to call the methods in any package, but classes calling your method have to be subclasses of your class.

However, what if you want to provide access only for classes in certain packages and don't want your class to be final to prevent subclassing? CheckPoint allows you to specify a list of package classes that can call your method.

```
public void methodAllowedFromManyClasses() {
 CheckPoint cp = new CheckPoint();
 cp.allowPackage("com.safepackage1");
 cp.allowPackage("com.safepackage2");
 cp.checkAccess();

//rest of the code for this method
}
```

If classes in the default package are supposed to be allowed access, the following can be used:

```
checkPoint.allowPackage(CheckPoint.DEFAULT_
PACKAGE);
```

What's the Catch?

Well, first of all, every call to your method will result in the execution of code in CheckPoint, which means that performance will be affected. I personally haven't seen a visible difference in applications using CheckPoint for one simple reason: most of the time I use it only for a couple of methods that are related to security.

Also, many would argue that just as C++'s friend is not object-oriented, the CheckPoint type of access control is not OOP friendly. I tend to disagree. However, you should make sure you're not breaking any OOP techniques by using CheckPoint.

Conclusion

Stack traces provide enough information about the thread's path to a line of code. We can use the contents of a trace to find out information about the caller and decide whether we want that method to be allowed access.

Author Bio

Ashutosh Narhari is a Java consultant at Ernst & Young. His primary interest is applying object-oriented techniques to create robust Web-based applications, and has been programming in Java since 1995.

ashutosh@narhari.com

Listing 1

```
private void parse() {
  // Store the name of this class and package
  String checkPointClassPlusPackage = this.getClass().getName();

  // Parse Stack Trace
  StringTokenizer st = new StringTokenizer(stackTrace, "\n");

  // Skip until we have hit the line containing this class's con-
structor
  // which created the throwable
  if (!st.hasMoreTokens()) {
   parseFailed = true;
   return;
  }
  String ignoredLine = st.nextToken();
  while(ignoredLine.indexOf(checkPointClassPlusPackage) == -1) {
   //System.out.println("Ignoring until <init>: " + ignoredLine);
   if (!st.hasMoreTokens()) {
    parseFailed = true;
    return;
   }
   ignoredLine = st.nextToken();
  }

  //Ignore next line [information about the method which instan-
tiated CheckPoint]
  if (!st.hasMoreTokens()) {
   parseFailed = true;
   return;
  }
  ignoredLine = st.nextToken();
  //System.out.println("Ignoring Caller info: " + ignoredLine);

  //Now we have reached the line containing caller's information
  if (!st.hasMoreTokens()) {
   parseFailed = true;
   return;
  }
  String lineToParse = st.nextToken();
  //System.out.println("lineToParse: " + lineToParse);



  //Get everything in between 'at ' and '('
  int idxAt = lineToParse.indexOf("at");
  if (idxAt == -1) {
   parseFailed = true;
   return; //failed to parse
  }
  int idxLeftParan = lineToParse.indexOf("(", idxAt+1);
  if (idxLeftParan == -1) {
   parseFailed = true;
   return; //failed to parse
  }
  String classPlusMethod = lineToParse.substring(idxAt+3,
idxLeftParan);

  //Parse package.class.method
  int lastDotIndex = classPlusMethod.lastIndexOf(".");
  if (lastDotIndex == -1) {
   parseFailed = true;
   return;
  }

  String fullClassName = classPlusMethod.substring(0,
lastDotIndex);
  this.callerMethodName =
classPlusMethod.substring(lastDotIndex+1);

  if (fullClassName.indexOf(".") > -1) {
   lastDotIndex = fullClassName.lastIndexOf(".");
   this.callerPackageName = fullClassName.substring(0,
lastDotIndex);
   this.callerClassName   =
fullClassName.substring(lastDotIndex+1);
  } else {
   this.callerPackageName = DEFAULT_PACKAGE;
   this.callerClassName = fullClassName;
  }

 }
```

Download the Code! www.JavaDevelopersJournal.com

A THREE-WAY QUERY

MORE THAN ONE PATH TO SUCCESSFUL DATA RETRIEVAL

WRITTEN BY TERESA LAU

Lightweight Directory Access Protocol (LDAP) is fast becoming a de facto access method for common directory information. XML is now the standard for data exchange on the Web. The relational database, a technology for modeling data in table form, has always been the most popular way to store and retrieve application data.

How do these three technologies come together, other than the fact that they're all buzzwords in the technology world? They're all a way to represent data, which you can easily access using the powerful set of query capabilities they each provide. LDAP data can be queried using LDAP directory search, XML data can be queried using XPath, and relational database data can be queried using SQL.

In this article I describe an Employee Information Application that can use all three methods to store and query data. You'll learn how to construct each kind of query and write Java code to retrieve data, respectively. It's of great value to you as a developer because you're likely to come across systems implemented using one of these technologies and be asked to develop applications on top of it.

The Employee Information Application

Figure 1 shows the user interface of a simple Employee Information Application, organizing employees by last name, first name, phone, and e-mail address.

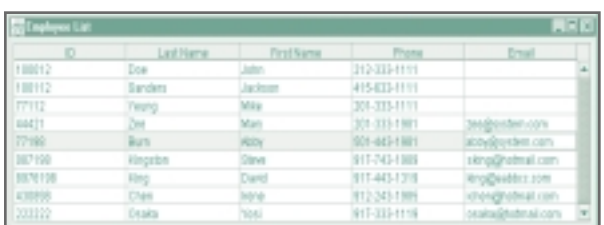


FIGURE 1 Employee information screen

Each employee can be represented as a Java object (seen below). In the next few sections, I discuss how this data can be represented and queried in various ways.

```
class Employee
{
private String id;
private String lastName;
private String firstName;
private String phone;
private String e-mail;
}
```

Relational Database and SQL

The relational database is a common way to model application data. You can model all the employee information described above into one table called Employee. The ID will be a unique key used to identify an employee.

```
create table Employee
{
 id  varchar(64)          not null;
 lastName varchar(64)     null;
 firstName varchar(64)    null;
 phoneNumber  varchar(12) null;
 e-mail  varchar(64)      null
}
create unique index IndxEmp on Employee (id)
```

To retrieve data from relational database, use SQL. Most developers are familiar with SQL, so I won't explain it here. A standard SQL looks like this:

```
Select <field1>,  <field2>
From <table>
Where <field1> = <value1>
```

```
And   <field2> = <value2>
And   …
```

Querying a Relational Database in Java Code

To access the database using Java, you can use JDBC, the standard API for executing SQL statements in any relational database. With JDBC, different drivers can be installed dynamically to access different databases. As a result, your code remains the same irrespective of the kind of database you're using. The only difference is you need to include that particular database driver class in your classpath when you run the program; and inside the code you need to load your specific database driver class at the beginning.

In my implementation I use Sybase for the database and Sybase jConnect as the JDBC driver. To get the code to run I put jConnect.jar in my classpath when I run my program. To log in to the database, I specify the login properties below, which I'll put into a properties file, "db.properties."

```
user=username
password=password
server=servername
sqlinitstring=use databasename
```

Now I'll initialize by creating a connection to the database using the code below:

```
Class.forName("com.sybase.jdbc.SybDriver");
Properties prop = Util.loadProperties ("db.proper-
ties");
String URL="jdbc:sybase:Tds:whitehorn:6600";
Connection connection =
DriverManager.getConnection(URL, prop);
```

This code does the following:

1. Calls Class.forName() to load the database driver (in this case, the Sybase driver)
2. Calls loadProperties (), a utility function I wrote (see Listing 1) to load the file db.properties to a Properties object
3. Sets up the URL to connect to the database (the documentation of your JDBC driver should tell you what to put in your URL; it's of the form jdbc: protocol: data source information.)
4. Passes URL and connection properties to getConnection() to create a database connection

Use this connection to run SQL statements to get all the employees from the table using the code below:

```
String sql ="select id, lastName, firstName, " +
                  "phone, e-mail  from Employee" ;
List employeeList = new ArrayList();
Statement stmt = connection.createStatement();
ResultSet s = stmt.executeQuery(sql);
while (s.next())
{
        String id = s.getString ("id");
        //   .. Get other fields e.g. lastName  simi-
larly
       Employee employee = new Employee(id);
       // … Set other fields into employee object sim-
ilarly
        employeeList.add(employee);
}
```

This code does the following:

1. Creates a statement, then executes the SQL to return a ResultSet
2. Iterates through the ResultSet, and for each row extracts the column value out and sets the appropriate field in the Employee object
3. Adds the Employee object to the EmployeeList

The full source of the database query code can be found in the DBImpl class in Listing 2.

LDAP and LDAP Directory Search

LDAP is a hierarchical way of storing data. It's commonly used by system administrators to store system and user information. LDAP stores data in a directory structure similar to the Unix file system. The basic unit of information in a directory is an entry. An entry has attributes, each of which has a type and one or more values. Figure 2 shows how employee information may be represented in LDAP.

In Figure 2, the top entry is an organization that has attributes dc=whitehorn, dc=com. Note that for the same attribute type dc, you can have more than one value. Under the organization entry you can have children, which are also entries. For example, location newyork and location toronto are both child entries of organization. To identify an entry uniquely, a DN (distinguished name) is used. DN is similar to the concept of a primary key in a database. In this example, the DN of John Doe is represented as:

```
Uid=100012, ou=Employee,
dc=newyork,dc=whitehorn,dc=com
```

Inside the John Doe entry you'll find other attributes such as lastName, firstName, phone, and e-mail.

An LDAP search operation is used to search a directory and retrieve individual directory entries. The LDAP search operation has eight parameters. I'll describe only three of them here, which should be sufficient for a simple query. (Refer to the links in the Resources section to learn more about LDAP searches.)

1. ***Search Base:*** This indicates the top of the tree where you want to start the search from. It's expressed in the form of a DN.
2. ***Attributes:*** Similar to the Select clause in the SQL, this is a

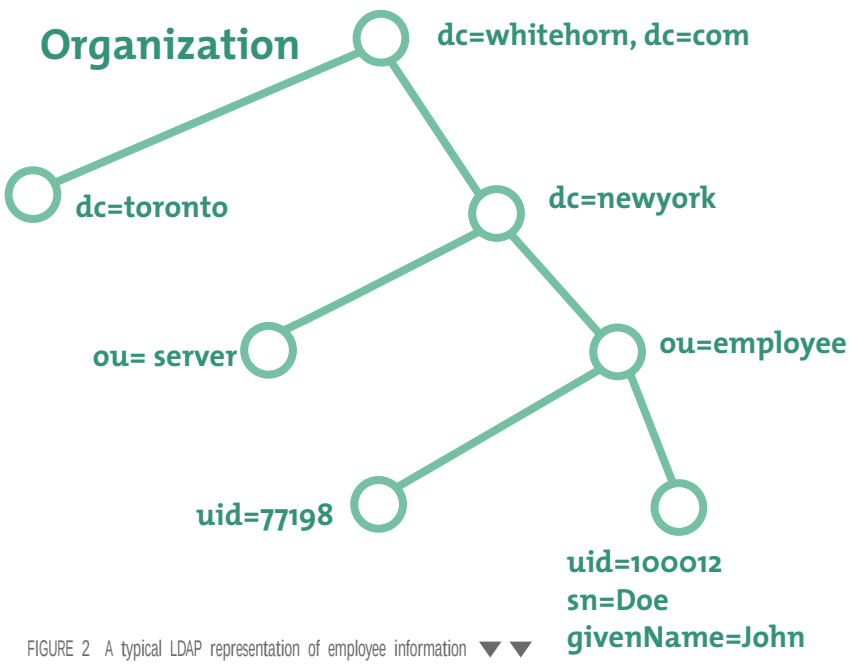


FIGURE 2 A typical LDAP representation of employee information

list of attributes to be returned for each matching entry.

3. ***Search Filter:*** An expression that describes the types of entries to be returned. It's similar to the Where clause in SQL. A filter can be in the following form:

```
Attr=value                  equal
Attr~=value                 approximate
Attr=*value*                use * to indicate wild card
                            any position
Attr>=value                 greater or equal
Attr<=value                 less or equal
Attr=*                      presence
(&(filter1)(filter2))       and
( | (filter1)(filter2))     or
( ! (filter))               negation
```

Querying LDAP In Java Code

In Java you can use Java Naming and Directory Interface s(JNDI) to do a directory search on the LDAP server. JNDI is an API that provides naming and directory functionality to applications written using Java. JNDI is included in the Java 2 SDK, v1.3, and later releases. If you're using earlier Java versions, you'll need to include the JNDI extension package in your classpath.

JNDI is defined to be independent of any specific directory service implementation. To use LDAP as the directory service implementation, add ldapbp.jar, ldap.jar, and providerutil.jar to your classpath to access the LDAP service provider classes. You can download the LDAP service provider release from http://java.sun.com/products/jndi/#download.

In my example, the LDAP server I used is Netscape Directory Server, but it really doesn't matter what LDAP server you're using: the code should still be the same.

To access the LDAP server I need to specify information on how to get to it. I put this information in the properties file "ldap.properties" as shown below. This information includes the name of the service class provider (in this case, the Sun LDAP service provider class), the URL to the LDAP server, and the user login and password to the LDAP server.

```
java.naming.factory.initial=com.sun.jndi.ldap.LdapCtx
  Factory
java.naming.provider.url=ldap:/localhost:1100/dc=ny,dc
  =whs,dc=com
java.naming.security.principal=user
java.naming.security.credentials=password
```

In my initialization code below, I load the properties file into a Properties object and then create an InitialDirConext:

```
Properties  p= Util.getProperties("ldap.properties"):
DirContext ctx = new InitialDirContext(p);
```

Now that I have an InitialDirContext, I can use it to look up all the employee entries using the code below:

```
String searchRoot = "ou=Employee" ;
String filter="(uid=*) ";
String[] attrIDs ={"uid", "givenname", "sn", "tele-
  phonenumber", "mail"};
SearchControls ctls = new SearchControls();
ctls.setReturningAttributes(attrIDs);

List employeeList = new ArrayList  ();
NamingEnumeration list =
          ctx.search(searchRoot, filter, new
SearchControls());
while (list.hasMore())
{
      SearchResult nc = (SearchResult)list.next();
      Attributes attrs = nc.getAttributes();
      Attribute attrId =  (Attribute)
attrs.get("uid");
      String id = (String) attrId.get();
      Employee e = new Employee(id);
       //  … similarly get other fields to fill out
       emplyeeList.add(employee);
}
```

This code does the following:

1. Specifies the three parameters for directory search: searchRoot, filter, and attributes
2. Using the initial context, does a search base on these conditions and returns a List of SearchResult
3. Iterates through the SearchResult, and for each Entry return retrieves its attributes and assembles the Employee object
4. Adds the Employee object to the employeeList

The complete LDAP search code can be found in LDAPImpl class in Listing 3.

XML and XPath

XML may not be the way to persist data, but with the B2B nature of applications, it's likely that we don't get our information directly from its persistence storage. For example, we could be getting the employee information from another Web service that will send us the information as an XML file, as shown in employees.xml in Listing 4.

XPath is a language used to query XML, commonly used inside an XSLT stylesheet for XML transformation. It describes how to locate specific elements (and attributes, processing instructions, etc.) in a document. Similar to the LDAP search, which has a search base, XPath has the concept of the "context node", which is where you start from. Everything you search is relative to the starting location.

I'll now show you some examples of an XPath query using the XML in Listing 4. The basic syntax of XPath is similar to file system addressing. You can use an absolute path by starting with/or a relative path. For example, /Employees/Employee/phone returns all the phone elements under Employees/Employee. Specifying * will get you all the elements located in the preceding path, so //Employee /* will return all elements (e.g., lastName, phone) under any Employee element in the document.

Expression in a square bracket can further specify an element. This is similar to a Where clause in a SQL. For example, Employee[starts-with(./lastName/text(),'D')] returns all employees with a lastName beginning in D. You can also specify the position of an element as in //Employees/Employee[2], which returns the second Employee child of Employees.

Attributes are specified by an @prefix. For example, //@city returns all the city attributes in the document. Attributes can also be used inside the square bracket to specify an element. An example is Addr[@city='newyork'], which returns all the Address elements that have an attribute city=newyork.

You can use a Boolean operation inside the square bracket. For example:

```
1. Employee[boolean(./lastName) or boolean (./e-mail)]
2. Employee[boolean(./lastName) and boolean (./e-
   mail)]
```

The first query represents an employee that has lastName or e-mail attributes, while the second query represents an employee that has both lastName and e-mail attributes. There are lists of functions for Node-Set, Boolean, Number, and String that you can use within an expression (see the Resources section).

You can apply more conditions in sequence by entering square brackets one after another. This is similar to a subquery in SQL. In addition, if the query is sequence-dependent, the result will depend on which condition you specify first. This can be seen in the example below:

```
1. //Employees/Employee[1][starts-
   with(./lastName/text(),'D') ]
2. //Employees/Employee[starts-
   with(./lastName/text(),'D') ][1]
```

Query 1 first selects all the first Employee elements under any Employees element, and then from that list, selects the ones that have a lastName beginning with "D." Query 2 first gets all the Employee elements under employees that have lastName D, and then from that list, selects the first one. The result and number of elements returned from these two queries may be different in this case.

There are many more powerful features of XPath that I haven't described here. Refer to the Resources section to learn more.

Querying XML in Java Code

To query XML using XPath, we can use the JAXP (Java API for XML Parsing) API. JAXP provides a common interface for creating and using the standard SAX, DOM, and XSLT APIs in Java, regardless of which vendor's implementation is actually being used. JAXP API 1.1. is included in the J2SE 1.4, or you can get it from http://java.sun.com/xml/download.html to be used as an optional package for JDK 1.1.8 and above.

In my example, I use the Xerces XML parser and Xalan XSLT processor, both from Apache (www.apache.org), but you can use any other parsers or XSLT processors that implement the JAXP API. In my classpath I need to include xerces.jar and xalan.jar so that the Apache classes can be found. The following code prepares for an XPath query by creating a DOM parser first. It uses the Xerces implementation of the DocumentBuilderFactory to create a Xerces DOM parser.

```
 DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
 domParser = factory.newDocumentBuilder();
```

Now that I have a DOM parser, I can use the following code to do an XPath query to get all the Employees elements.

```
URL cfgURL = Util.createURL("employees.xml");
Document d = domParser.parse(cfgURL.toString());

String XPath = "//Employee
List employeeList = new ArrayList();
NodeList nl = XPathAPI.selectNodeList(d, XPath );
for (int i = 0; i< nl.getLength(); i++)
{
     Node n = nl.item(i);
      // id is an attribute
     Node nm = n.getAttributes().getNamedItem("id");
     Employee emp = new Employee (nm.getNodeValue());

     // Other fields are elements
      NodeList childNodes = n.getChildNodes();
      Node lastname = lookup (childNodes, "lastName");
      // …do similarly for other fields

emp.setLastName(lastname.getFirstChild().getNodeValue(
));
      employeeList.add (emp);
}
```

This code does the following:

1. Creates a DOM document from the XML file employees.xml (see Listing 4) by parsing the XML file
2. Calls org.apache.xpath.XPathAPI.selectNodeList (from Xalan API) to run a query on the document using the XPath query statement
3. Iterates the result (NodeList) to assemble the Employee object; (note that in this XML, ID is an attribute of Employee and it's retrieved in a different way than the other fields, e.g., lastName or firstName, which are elements of Employee)
4. Adds the Employee object to the EmployeeList

The full listing of the XPath search code can be found in the XMLImpl class in Listing 5.

A Mode Complicated Query

To further illustrate the various ways to query the data, I'll now explain how to assemble the respective queries based on the same search condition as shown below:

```
1. (Lastname begins with 'D%'  OR firstname begins
  with 'M%')      AND
2. Does not have an e-mail attribute
```

The SQL query for this search is:

```
select id, lastName, firstName, phone, e-mail  from
Employee
```

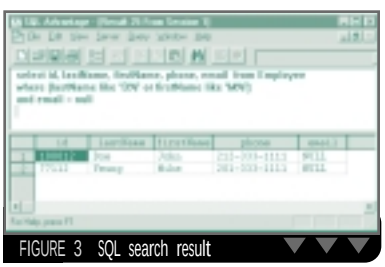


FIGURE 3 SQL search result

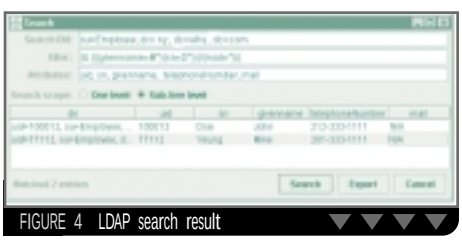


FIGURE 4 LDAP search result

FIGURE 5 XPath search result

```
where (lastName like 'D%' or firstName like 'M%')
and  e-mail =  null
```

Figure 3 shows the result of running this query in a SQL tool. The LDAP query for this search is:

```
Search Base:      ou=Employee,dc=newyork,  dc=white
                  horn,dc=com
Search Filter:    (& (|(givenname=M*)(sn=D*))
                  (!(mail=*)))
Attributes:       uid, sn, givenName, telephoneNumber,
                  mail
```

The result of running this query in the query tool of an LDAP browser is shown in Figure 4.

The XPath query for this search is:

```
//Employee[starts-with(./lastName/text(),'D') or
starts-with(./firstName/text(),'M')]
   [not(boolean(./e-mail/text()))]
```

Figure 5 shows the result of running this in XPath using a tool called Cooktop XML editor (a free XML editor that can be downloaded at www.xmleverywhere.com/cooktop).

You can easily put the queries we constructed here back into the code we wrote in the last section, and have the results returned programatically in Java.

Common Interface

Depending on where the data is stored or comes from, the implementation for the Employee Search Application will be different. We could apply the Bridge pattern from the book, *Design Patterns,* to write the client for the Employee Search Application. The Bridge pattern separates a class interface from its implementation, so you can vary or replace the implementation without changing the client code. To do so I define the following interface:

```
Interface EmployeeQuery
{
   // Does initialization e.g. connect to server
   init();

   // Generate query based on Condition and return
     List of Employee
   List search  (Condition c);
}
```

The client can then call methods in this interface without knowing what the underlying implementation is like. In my test client program, I create a method call runTest that calls init() and then does a search with empty conditions. This returns a list of all employees, printing their first and last names.

```
public static void runTest(EmployeeQuery eq)
{
    System.out.println ("Running class " +
eq.getClass().toString());
    eq.init();
    List l = eq.search (new Conditions());
    Iterator i = l.iterator();
    while (i.hasNext())
    {
          Employee e = (Employee) i.next();
          System.out.println (e.getId() + ":"  +
e.getFirstName() + " " + e.getLastName() );
    }
}
```

Then I create the database, XML and LDAP implementations of EmployeeQuery, and call runTest on them, respectively, to test out the implementation.

```
public static void main(String[] args)
{
     runTest(new DBImpl());
     runTest(new XMLImpl());
     runTest(new LDAPImpl());
}
```

You can find the complete source code of DBImpl, LDAPImpl, and XMLImpl shown, respectively, in Listing 2, Listing 3, and Listing 5. Because this is only an example, currently all they do is return all the employees in their system. It doesn't use the condition parameter passed in to generate the query. You can do more work to specify conditions from a GUI, which you turn into the respective query to run and return the matching items.

Conclusion

I've shown you three ways of implementing the same client based on how the data is stored. It's unlikely that you'll have to implement an application in all three different ways. The purpose of this article is to show you the different syntaxes and code APIs you can use, so you can easily tackle each kind of query with minimal effort when you need to. I've only touched briefly on SQL, LDAP, and XPath. Each of them is in itself a big subject that you can study in detail when you have time.

A final point to bear in mind when developing your client: it's possible that the underlying implementation may change or that your client may be used with more than one implementation. In that case, your system will benefit if you use an interface when designing your code.

Resources

Relational databases and SQL:

1. *JDBC API Doc:* http://java.sun.com/j2se/1.3/docs/api/java/sql/package-summary.html

LDAP:

2. *JNDI API Doc:* http://java.sun.com/products/jndi/java-doc/index.html
3. *Excerpts from an LDAP book:* http://developer.netscape.com/docs/books/macmillan/ldap/ldapbk.html
4. *Sun JNDI tutorial:* http://java.sun.com/products/jndi/tutorial/TOC.html

XML and XPath:

5. *JAXP API Doc:* http://java.sun.com/xml/jaxp-1.1/docs/api/
6. *Official W3C site:* www.w3.org/TR/XPath
7. *Quick Reference Card:* www.mulberrytech.com/quick-ref/index.html
8. *XPath example:* www.zvon.org/xxl/XPathTutorial/General/examples.html

Author Bio

Teresa Lau has been an independent Java consultant for over four years. Her emphasis is in financial applications and her clients include major financial institutions in New York and Toronto. She received her MS in computer science from the University of Waterloo, and her BS in engineering from the University of California, Berkeley.

ttylau@whsys.com

Listing 1

```
import java.util.*;
import java.io.*;
import java.net.URL;
import java.net.MalformedURLException;

public class Util
{

  static  public URL createURL(String name)
  {
    // Turn a URL-string or filename into a URL object
    URL url = null;
    try
    {
       url = new URL(name);
       return url;
    }
    catch (MalformedURLException ex) {;}

     // If it wasn't a URL, it must be a file...
    try
    {
       File file = new File(name);
       url  = new URL("file:" + file.getAbsolutePath());
       return url;
    }
    catch (MalformedURLException ex)
    {
       System.out.println ("Malformed URL " );
       return null;
    }
  }

  public static Properties loadProperties(String fileName)
  {
     try
     {
       FileInputStream fis = new FileInputStream(fileName);
       Properties properties = new Properties();
       properties.load(fis);
       return properties;
     }
     catch(FileNotFoundException e)
     {
       System.out.println ("File not found " + fileName);
       return null;
     }
     catch(IOException e)
     {
       System.out.println ( e.getMessage());
       return null;
     }
   }
}
```

Listing 2

```
import java.util.*;
import java.sql.*;

public class DBImpl implements EmployeeQuery
{
  private Connection connection;
  private String URL="jdbc:sybase:Tds:whitehorn-dev:6600";
  private String PROPERTIES="db.properties";
  private String DRIVER="com.sybase.jdbc.SybDriver";

  public void init()
  {
    try
    {
      Properties p = Util.loadProperties(PROPERTIES);
      Class.forName(DRIVER);
      connection = DriverManager.getConnection(URL, p);
    }
    catch (ClassNotFoundException e)
    {
      System.out.println ("Cannot find Sybase driver ");
    }
    catch (SQLException e)
    {
      System.out.println ("Error logging into database ");
    }
  }

  public java.util.List search(Conditions c)
  {
     String sql = "select id, lastName, firstName, " +
                  "phone, e-mail  from Employee" ;
     try
     {
       Statement stmt = connection.createStatement();
       ResultSet s = stmt.executeQuery(sql);
       List employeeList = new ArrayList();
       while (s.next())
       {
         Employee emp = new Employee (s.getString("id"));
         emp.setLastName (s.getString("lastName"));
         emp.setFirstName(s.getString("firstName"));
         employeeList.add(emp);
       }
       return employeeList;
     }
     catch (SQLException e)
     {
       e.printStackTrace();
       return null;
     }
  }
}
```

Listing 3

```
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;

public class LDAPImpl implements EmployeeQuery
{
  private String PROPERTIES="ldap.properties";
  private DirContext ctx = null;

  public void init()
  {
    try
    {
       Properties p = Util.loadProperties(PROPERTIES);
       ctx = new InitialDirContext(p);
    }
    catch(NamingException e)
    {
       System.out.println ("Failed getting context "  );
    }
  }

  public java.util.List search(Conditions c)
  {
    List employeeList  = new ArrayList();

    String searchRoot = "ou=People";
    String filter = "(uid=*)";
    String[] attrIDs = {"uid", "givenname", "sn",
 "telephonenumber", "mail"};
    SearchControls ctls = new SearchControls();
    ctls.setReturningAttributes(attrIDs);

    try
    {
      NamingEnumeration list =
                  ctx.search(searchRoot, filter, ctls);

      while (list.hasMore())
      {
         SearchResult nc = (SearchResult)list.next();
         Attributes attrs = nc.getAttributes();
         Attribute attrId =  (Attribute) attrs.get("uid");
         String id = (String) attrId.get();
         Employee emp = new Employee(id);

         Attribute attFN=(Attribute)attrs.get("givenname");
         String firstName=(String) attFN.get();
         emp.setFirstName(firstName);

         Attribute attLN=(Attribute)attrs.get("sn");
         String lastName=(String) attLN.get();
         emp.setLastName(lastName);

  employeeList.add(emp);
      }
      return employeeList;
    }
    catch (NamingException e)
    {
       System.out.println ( e.getMessage());
       return null;
    }
  }
}
```

Listing 4

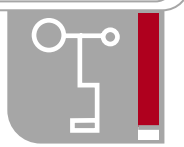
```
<?xml version='1.0'?>
<Employees>
  <Employee id='100012' >
    <lastName>Doe</lastName>
    <firstName>John</firstName>
    <phone>212-333-1111</phone>
  </Employee>
  <Employee id='100112' >
    <lastName>Sanders</lastName>
    <firstName>Jackson</firstName>
    <phone>415-633-1111</phone>
  </Employee>
  <Employee id='77112' >
    <lastName>Yeung</lastName>
    <firstName>Mike</firstName>
    <phone>201-333-1111</phone>
  </Employee>
  <Employee id='44421' >
    <lastName>Zee</lastName>
    <firstName>Mary</firstName>
    <phone>201-333-1981</phone>
    <e-mail>zee@system.com</e-mail>
  </Employee>
  <Employee id='77198' >
    <lastName>Burn</lastName>
    <firstName>Abby</firstName>
    <phone>501-443-1981</phone>
    <e-mail>abby@system.com</e-mail>
  </Employee>
```

```
  <Employee id='087198' >
    <lastName>Kingston</lastName>
    <firstName>Steve</firstName>
    <phone>917-743-1989</phone>
    <e-mail>sking@hotmail.com</e-mail>
  </Employee>
  <Employee id='8976198' >
    <lastName>King</lastName>
    <firstName>David</firstName>
    <phone>917-443-1319</phone>
    <e-mail>king@aabbcc.com</e-mail>
  </Employee>
  <Employee id='430898' >
    <lastName>Chen</lastName>
    <firstName>Irene</firstName>
    <phone>912-243-1985</phone>
    <e-mail>ichen@hotmail.com</e-mail>
  </Employee>
  <Employee id='222222' >
    <lastName>Osaka</lastName>
    <firstName>Yosi</firstName>
    <phone>917-333-1119</phone>
    <e-mail>osaka@hotmail.com</e-mail>
  </Employee>
</Employees>
```

Listing 5

```
import java.util.*;
import java.net.URL;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.apache.xpath.*;


public class XMLImpl implements EmployeeQuery
{
  private String XML_FILE="employees.xml";
  private DocumentBuilder domParser;

  public void init()
  {
    try
    {
       DocumentBuilderFactory factory =
                DocumentBuilderFactory.newInstance();
       domParser = factory.newDocumentBuilder();
    }
    catch (Exception e)
    {
       System.out.println ("Failed creating dom parser ");
       e.printStackTrace();
    }
  }
   /* Return a node from the nodelist which has name */
  private Node lookup (NodeList nl, String name)
  {
    for (int i = 0; i< nl.getLength(); i++)
    {
       Node n = nl.item(i);
       if (n.getNodeName().equals(name))
          return n;
    }
    return null;
  }

  public java.util.List search(Conditions c)
  {
    try
    {
       URL cfgURL = Util.createURL(XML_FILE);
       Document d = domParser.parse(cfgURL.toString());
       String xpath = "//Employee";
       List employeeList = new ArrayList();
       NodeList nl = XPathAPI.selectNodeList(d, xpath );
       for (int i = 0; i< nl.getLength(); i++)
       {
          Node n = nl.item(i);

          // id is an attribute
          Node nm = n.getAttributes().getNamedItem("id");
          Employee emp = new Employee(nm.getNodeValue());

          // Other fields are elements
          NodeList childNodes = n.getChildNodes();
          Node lname = lookup (childNodes, "lastName");
      String ln = lname.getFirstChild().getNodeValue();
          emp.setLastName(ln);
      Node fname = lookup (childNodes, "firstName");
      String fn = fname.getFirstChild().getNodeValue();
      emp.setFirstName(fn);
      employeeList.add (emp);
       }
     return employeeList;
     }
     catch (Exception e)
     {
        e.printStackTrace();
        return null;
     }

  }
}
```

Download the Code! www.JavaDevelopersJournal.com



Prescriptions for Your Java Ailments

I'M DEVELOPING AN INTERNET APPLICATION WHERE THE USER CAN REGISTER. HOW WOULD I CHECK IF THE E-MAIL ADDRESS ENTERED IS VALID?

I have a couple of ideas you can implement. First, I suggest using JavaScript in the browser to check for allowable characters by scanning the input. The JavaScript could also look for a single instance of the @ character and at least one period. For a list of all allowable characters go to www.faqs.org/rfcs/rfc822.html.

Now for the server side. Make sure (1) the e-mail address is syntactically correct, and (2) the supplied domain name is valid. The reason is that we can't ascertain whether an e-mail address is truly valid until we send e-mail to it. Let's take a look at some sample code that performs the JavaScript equivalent on the server.

```
import javax.mail.internet.*;

public boolean isValidEmail (String email) {

   boolean emailOK = true;

    // Use javax.mail.internet.InternetAddress to
validate email format

   try {
                InternetAddress addr = new
InternetAddress( email );
   }
   catch ( AddressException ae) {
      emailOK = false;
   }
   return emailOK;
}
```

This routine performs a simple format check of the supplied e-mail address. We could also take the domain name and do a simple lookup following the same principle. You would use a StringTokenizer to parse out the host portion of the supplied e-mail address and pass it to the isValidHost method. Here's how it can be done:

```
import java.net.*;
import java.util.*;
public boolean isValidHost ( String host ) {
      boolean hostOK = true;
      try {
     InetAddress addr = InetAddress.getByName(
host ) ;
   }
   catch ( UnknownHostException uhe ) {
      hostOK = false;
   }
   return hostOK;
}
```

If the host is valid and the e-mail address is in the right format, you should be all set. To perform deeper validation could be time-consuming and could slow down your application. We could get more elaborate and use the JNDI provider for DNS as well as check to see whether the supplied domain name has a valid MX address. We could also open a socket connection to the server that has the appropriate MX record for the domain and send it RCPT TO and VRFY commands, but this would be slow and isn't recommended.

I'D LIKE TO CONVERT AN XML DOCUMENT INTO A PDF. CAN YOU TELL ME THE BEST WAY?

The Apache project has the right package for you – see http://xml.apache.org/fop/index.html. FOP is a print formatter that takes advantage of XSL formatting objects. Essentially, it reads the formatting object tree and then turns it into a PDF document. The formatting object tree can be in the form of an XML document. You can pass it in memory as a DOM document using Xalan/Xerces or via SAX events.

WE'RE DOING SOME XML PARSING USING THE JAXP.JAR AND CRIMSON.JAR, AND THE SEALING VIOLATION OCCURS WHEN WE RUN CODE AS AN EJB BUT NOT AS A STAND-ALONE JAVA APPLICATION. COULD YOU EXPLAIN WHAT A SEALING VIOLATION IS?

Sealing violations occur when you try to load different classes from the same page from different libraries. A sealed package is contained within a library whose manifest specifies that all classes for that package must be loaded from the same library. This typically occurs when you have more than one DOM implementation on your classpath.

Move the crimson.jar and jaxp.jar to the first elements on the classpath and your problem should disappear.

IS THERE ANY ADDITIONAL OVERHEAD IN USING IMPORT JAVA.UTIL.* COMPARED TO IMPORT JAVA.UTIL.VECTOR?

I run across this type of question frequently. The answer is simple but will require some explanation. When you use imports in the form of java.util.Vector, it's referred to as a single type import. When you use imports in the form of java.util.*, it's referred to as an import on demand. Both types of imports are passive by nature. The Java compiler will load imported classes only when the class is actually used, not when it's simply imported. This is a hint to the compiler to tell it where to locate the class. The Java bytecode doesn't contain any references to the import statement. In other words, there's no additional overhead.

My bias states that it's bad practice to use import on demand. I believe that developers should list each imported class they intend to use. This helps your peers as it provides additional documentation on what the program is actually using. The rule of thumb I recommend is that if you're importing four or fewer classes from a particular package, you should use single type import.

I KNOW HOW TO ACCESS AN HTTP-BASED URL USING JAVA.NET.URL. HOW DO I ACCESS AN HTTPS-BASED URL?

The prescription for your problem is to use the Java Secure Sockets Extension (JSSE) package, which you can download from http:\\java.sun.com and install on your classpath. If you try to use the URL connection object without this package, you'll receive MalformedURLException. You'll have to set a couple of system properties

programmatically and add a new security provider. Here's the code:

```
System.setProperty("java.protocol.handler.pkgs"
, "com.sun.net.ssl.internal.www.protocol");
Security.addProvider(new com.sun.net.ssl.inter-
nal.ssl.Provider());
```

After setting the security provider, you can then use:

```
URL url = new URL("https://doctorjava");
```

There are a couple of issues with using this approach. If the URL you're connecting to has either an invalid or an unsigned certificate, then the URL connection will throw an SSL Exception. You'll still be able to use the input stream. The example above assumes that SSL is on the default port of 443. For URLs where you're not using a nondefault port, you can simply append it as follows:

```
URL url = new URL("https://doctorjava:7002");
```

I'M CURRENTLY USING JAVA.NET.URL TO ACCESS AN HTTP-BASED URL BUT KEEP GETTING AN AUTHENTICATION CHALLENGE FROM MY PROXY. CAN YOU POINT ME IN THE RIGHT DIRECTION?

This question is perfectly timed. A fellow colleague asked me the same one the other day so I didn't have to do any research on the answer. I'll outline a solution that works on JDK 1.2 and above and will encourage those still running an older VM to upgrade.

The first symptom you'll notice is getting a FileNotFoundException thrown at you when you try to read the associated InputStream. Java provides a mechanism that allows us to pass authentication credentials as part of the java.net package. Let's discuss the Authenticator class.

When a Web or proxy server requires authentication, the Authenticator's getPasswordAuthentication() method is invoked. You can return a PasswordAuthentication instance that contains the appropriate username and password. It's really simple. Here's some code that demonstrates this functionality:

```
// Install a new Authenticator
Authenticator.setDefault(newNetegrityAuthentica
tor());
 // Subclass Authenticator
 class NetegrityAuthenticator extends
     Authenticator {
     protected PasswordAuthentication
     getPasswordAuthentication() {
return newPasswordAuthentication
("uid","pass");
   }
 }
```

The values uid and pass in this example are hard-coded, but you could retrieve them programmatically from any source. The code would be typically invoked before your URL call. It really is this simple.

I NEED A METHOD THAT WILL RECURSIVELY LIST ALL FILES IN A DIRECTORY. CAN YOU HELP ME OUT?

Here's a simple routine that should do the trick:

```
public static void recurseFiles(Vector list, File
root) {
   if (root.isFile()) {
      list.addElement(root);
      return;
   }

 File[] files = root.listFiles();
 for (int i=0; i < files.length; i++) {
                if ( files[i].isFile() &&
files[i].getName().endsWith(".doc")) {
      list.addElement(files[i]);
      }
   if ( files[i].isDirectory()) {
      recurseFiles(list,files[i]);
   }
 }
```

• • •

I would like to conclude this column with my sincere condolences to the families and friends of the victims of the horrific tragedies that occurred in New York and Washington, DC. My heartfelt prayers are with you all.

• • •

Send your questions, praise, comments, and admiration to doctorjava@sys-con.com.

Published letters will be edited for length and clarity. Any reference to third parties or third-party products should not be construed as an endorsement by Doctor Java or Java Developer's Journal.

Author Bio

Doctor Java, a.k.a. James McGovern, moonlights as an enterprise architect with Hartford Technology Services Company , L.L.C. (www.htsco.com), an information technology consulting and services firm dedicated to helping businesses gain competitive advantage through the use of technology.

doctorjava@sys-con.com

JASON BRIGGS J2ME EDITOR

The Poor Cousin

A big date in the European gamer's calendar is the European Consumer Trade Show (ECTS) in London. It's a time for 16-year old boys to polish up their fake IDs, dust off the letters confirming their internship at a large game publisher (printed earlier in the week on letterhead paper sold at exorbitant rates by some enterprising individual), and steal their older sister's eyeliner to provide the beard-fluff on their faces with a bit of added support – so they can try to sneak into the show.

By all reports ECTS is like a poor cousin to the Electronic Entertainment Expo (E3) in the U.S., and while that's perhaps being a bit uncharitable, both Sony and Nintendo chose not to attend the show this year – indeed, Nintendo decided to hold its own event at another venue.

I was attending in the hope of seeing some inkling of the Java gaming industry that I'm sure is just around the corner. In terms of the mainstream industry, perhaps I'm becoming jaded as I get older, but I didn't see a hell of a lot of originality there. Sure, there were some interesting-looking 3D Massive Multiplayer Role Playing Games (MMRPGs) with graphics that, at times, defied belief, but there were also the 2D strategy games, the occasional platformer, the FPS shooter; games that didn't add anything new to their respective genres.

Not to tar the entire industry with the same brush, but in these cases, at the very least maybe a technological paradigm shift might inject some much-needed creativity into the products. In a number of cases I didn't see anything that couldn't have been done in Java – which made me wonder why they hadn't been...?

On the embedded front, there was very little at ECTS to interest a Java developer. The first of two exceptions was FUNCOM (www.funcom.com), which had an application to convert and transfer any photos from a PC to a mobile phone. The site was completely Java-driven, and a Java applet was used to manipulate the picture (changing the optimization and dithering, etc.) before sending it from the server using reverse-billing SMS. The only problem I noted with their application was that there didn't seem to be a way to make a less-than-photogenic developer look good when his photo was sent to the mobile. Sigh...

The other exception was a Korean-based company called Com2uS (www.com2us.com/www.phonegame.net), which develops games for a number of mobile platforms including J2ME (KVM), Game Virtual Machine (GVM by Sinjisoft), WAP, and BREW. Their games include arcade, puzzle, board, and casino genres, as well as role-playing. At the company's booth were a number of color mobile phones. Unfortunately, the woman from Com2uS was rather too attentive, otherwise their display might have wound up lacking a few devices...and half of their software catalog.

Com2uS is apparently one of the founding members of the Java Wireless Developers Council for Sun and a consultant to LG Telecom's EZ-Java program. In any case not only are the phones impressive to look at, but the games look brilliant.

So, apart from the aforementioned companies, not altogether a positive showing for Java at the show – at least from my point of view. Perhaps you were there as well? Maybe you noticed other Java and/or mobile Java games on display that I missed? Let me know...unless, of course, you were one of the large group of males reexperiencing their adolescence as they clustered around three bikini-clad models at a booth promoting an erotic game of some sort (of course I noticed...what am I, blind??). In which case, don't bother. You've probably fried a few too many brain cells to remember much else about the show.

jasonbriggs@sys-con.com

AUTHOR BIO

Jason Briggs works as a Java analyst programmer in London. He's been officially developing in Java for three years – unofficially for just over four.

J2ME INDEX

Some of the more commonly asked questions on the various forums for J2ME seem to be, "What is J2ME?" and "Is <so-and-so-product> a part of J2ME?" Here is where you will find all the APIs that fall beneath J2ME's umbrella, and the packages you will find within those APIs.

CONNECTED, LIMITED DEVICE CONFIGURATION (CLDC) – VERSION 1.0

java.io	input and output through data streams
java.lang	fundamental classes
java.util	collections, data and time facilities, other utilities
javax.microedition.io	generic connections classes

You can find more information on CLDC at the following URL:
http://java.sun.com/products/cldc/

CONNECTED DEVICE CONFIGURATION (CDC) – VERSION 0.2

java.io	input and output
java.lang	fundamental classes
java.lang.ref	reference object classes
java.lang.reflect	reflective information about classes
java.math	BigInteger support
java.net	networking support
java.security	security framework
java.security.cert	parsing and management of certificates
java.text	used for handling text, dates, numbers and messages
java.text.resources	contains a base class for locale elements
java.util	collections, date/time, miscellaneous functions
java.util.jar	reading Jar files
java.util.zip	reading Zip files
javax.microedition.io	connections classes

Look for more CDC information here:
http://java.sun.com/products/cdc/

MOBILE INFORMATION DEVICE PROFILE – VERSION 1.0

java.io	
java.lang	CLDC, plus an additional exception
java.util	CLDC, plus timer facilities
javax.microedition.io	networking support based upon the CLDC framework
javax.microedition.lcdui	for user interfaces for MIDP applications
javax.microedition.rms	persistent data storage
javax.microedition.midlet	defines applications and interactions between app and environment

The products page for MIDP is here:
http://java.sun.com/products/midp/

FOUNDATION PROFILE – VERSION 0.2

java.io	see CDC
java.lang	see CDC
java.lang.ref	see CDC
java.lang.reflect	see CDC
java.math	see CDC
java.net	see CDC
java.security	see CDC
java.security.cert	see CDC
java.security.acl	access control lists
java.security.interfaces	interfaces for generating keys
java.security.spec	key specifications, and algorithm parameter specifications
java.text	see CDC
java.text.resources	see CDC
java.util	see CDC
java.util.jar	see CDC
java.util.zip	see CDC
javax.microedition.io	see CDC

The profile products page is here:
http://java.sun.com/products/foundation/

J2ME RMI PROFILE (JSR #66)

This profile interoperates with J2SE RMI, and provides Java platform–to–Java platform remote method invocation for Java devices.

J2ME GAME PROFILE (JSR #134)

This is a proposed Micro Edition specification, so nothing is yet defined. According to the JCP home page for JSR #134 (the Game Profile), the following areas will be covered:

1. 3D Modeling and Rendering for Games
2. 3D Physics Modeling for Games
3. 3D Character Animation for Games
4. 2D Rendering and Video Buffer Flipping for Games
5. Game Marshalling and Networked Communication
6. Streaming Media for Games
7. Sound for Games
8. Game Controllers
9. Hardware Access for Games

PDA PROFILE (JSR #75)

The PDA Profile will provide UI and storage APIs for small, resource-limited handheld devices.

PERSONALJAVA SPECIFICATION – VERSION 1.2A

java.applet	full support from JDK1.1.8
java.awt	modified from JDK1.1.8
– note: there is an extra method for PJ for double-buffering in java.awt.Component	
java.awt.datatransfer	full support
java.awt.event	full support
java.awt.image	full support
java.awt.peer	modified
java.beans	full support
java.io	modified
java.lang	modified
java.lang.reflect	modified
java.math	optional – may or may not be supported
java.net	modified
java.rmi	optional
java.rmi.dgc	optional
java.rmi.registry	optional
java.rmi.server	optional
java.security	modified
java.security.acl	unsupported
java.security.cert	some classes required, some optional
java.security.interfaces	required if code signing is included
java.security.spec	required if code signing is included
java.sql	optional
java.text	full support
java.text.resources	modified
java.util	modified
java.util.jar	required if code signing is included
java.util.zip	modified

Additional PersonalJava specific packages are:

com.sun.awt	for mouseless environments
com.sun.lang	a couple of error & exception classes
com.sun.util	for handling timer events

PersonalJava will eventually be superseded by the Personal Profile. For more information on the PersonalJava Application Environment:
http://java.sun.com/products/personaljava/

JAVA TV – VERSION 1.0

javax.tv.carousel	access to broadcast file and directory data
javax.tv.graphics	root container access and alpha blending
javax.tv.locator	referencing data and resources
javax.tv.media	controls and events for management of real-time media
javax.tv.media.protocol	access to generic streaming data in a broadcast
javax.tv.net	IP datagram access
javax.tv.service	service information access
javax.tv.service.guide	supporting electronic program guides
javax.tv.service.navigation	services and hierarchical service information navigation
javax.tv.service.selection	select a service for presentation
javax.tv.service.transport	information about transport mechanisms
javax.tv.util	creating and managing timer events
javax.tv.xlet	communications interfaces used by apps and the app manager

Get off that couch and check out the JavaTV page at the following URL:
http://java.sun.com/products/javatv/

JAVA EMBEDDED SERVER – VERSION 2.0

com.sun.jes.service.http	servlet/resource registrations
com.sun.jes.service.http.auth.basic	http basic authentication
com.sun.jes.service.http.auth.users	management of users and their access
com.sun.jes.service.timer	for handling timer events
org.osgi.framework	consistent model for app. dev., supports dev. and use of services
org.osgi.service.device	detection of devices
org.osgi.service.http	http access of resources
org.osgi.service.log	logging facility

You can find more information on Embedded Server on the following site:
http://www.sun.com/software/embeddedserver/

JAVA CARD – VERSION 2.1.1

java.lang	fundamental classes
javacard.framework	core functionality of a JC Applet
javacard.security	security framework
javacardx.crypto	extension package with security classes and interfaces

Next time you use that American Express Blue card, you may want to know how it works, so take a look here:
http://java.sun.com/products/javacard/

Written by Mark Quinn and Julien Tranchant

Goodbye Snake

An introduction to game development for MIDP handsets

As time goes by, J2ME is the buzzword that's appearing in more and more magazines and talked about beside more office coffee machines than any other. And slowly, the big handset manufacturers are releasing their first MIDP-compliant phones into the marketplace. In previous issues of *JDJ* (Vol. 6, issues 7–9), you may have read Jason Briggs' articles introducing general MIDlet development techniques. This article will take a swift sidestep and focus on MIDP as a platform for developing games. We can assure you, in the near future, you won't be firing up Snake the next time your train is late.

Tied to the '80s

In the computer industry we're all becoming accustomed – and complacent – to the fact that processors get faster, memory capacity increases, disks become more capacious, and applications, as a result, become increasingly bloated. Most triple-A game titles released for the PC and console market now feature incredibly immersive three-dimensional environments and meticulously rendered full-motion video, and today's gamer typically demands this of his or her next purchase. The downside is that not only do these titles carry a development price tag upwards of $3 million and require two years of sweat (sometimes blood) and tears from a large development team, they also require a computer with impressive processing and graphics capabilities to run them. In comparison to these behemoths, the average MIDP device has a small, limited-color screen, very little RAM – often less than a megabyte – restricted amounts of data storage, and worst of all, a processor with only a few MHz.

Some may see this as a major flaw in the mobile gaming market; however, we strongly disagree. It might have seemed that the days when the bored teenage bedroom coder could cobble together a Manic Miner or Skool Daze were truly behind us – but not anymore. J2ME has taken us straight back to 1983; an era when the only thing that counted was pure, hard-core playability. Only this time we can get our fix where we want – and if you're really lucky, you may even get mistaken for tapping out a quick text-message to your boss!

Goals

In this article we take you through some of the basic concepts of game architecture, producing sample MIDP code as we go, and we'll discuss the salient features of the various libraries. We don't aim to produce a complete game, but by the end of the article you should understand what's needed to produce a simple game of your own and have some skeleton code upon which to base your work.

Getting Ready

As with the previous ***JDJ*** articles on MIDP, we'll use Sun's J2SE SDK1.3 and the J2ME Wireless Toolkit (WTK). If you don't have them already, go to http://java.sun.com/j2se/1.3/ or http://java.sun.com/products/j2mewtoolkit/ to download the latest versions.

Although we won't be working with it specifically in this article, also worth a mention is Sun's Forte for Java, an IDE that tightly integrates itself with the WTK. The Community Edition is available for a free download at www.sun.com/forte/ffj/buy.htm. If you do use Forte, be sure to install it before you install the WTK.

For the purposes of these examples, we'll assume a couple of standard directories, namely:

- ***C:\jdk1.3***: The root directory for the JDK
- ***C:\j2mewtk:*** The root directory for the Wireless Toolkit

Creating the Project

For this project we'll work with KToolbar. It's a small tool that comes standard with the WTK and gives a helpful interface to

Goodbye Snake

An introduction to game development for MIDP handsets

some of MIDP's peculiarities. KToolbar manages all the stages of compilation, preverification, JAR construction, and JAD file management for you. It also helps keep your source code organized by imposing a directory structure to work within. A quick word of warning: it's a useful tool, but not very intelligent. Be sure to check things by hand if you get yourself in a pickle.

The WTK will have installed itself into your start menu as the J2ME Wireless Toolkit. In that group will be a shortcut to KToolbar. Launch it and press the New Project button.

Enter both the Project Name and the MIDlet Class Name as "MyGame" and click on Create Project. Click OK on the settings box that appears. The defaults are fine.

You should see the output as in Figure 1. Your project has been created under the apps directory of the WTK installation.

Three directories , shown in Figure 2, will have been created for you:

- ***bin:*** The JAD and JAR files will be created here when you build your project.
- ***src:*** This is where you place your Java source files. KToolbar will compile anything under here with javac and add the compiled classes into the distribution JAR file in the bin directory.
- ***res:*** Put any resource files you use in your project in this directory. They will be added to the JAR file. You should not use the res/ prefix when referencing these files in your code, e.g., a file .../MyGame/res/images/myImage.png should be referenced in your MyGame Java code as /images/myImage.png.

> "At the heart of every game is the main game loop... a basic series of procedures for getting the player's input, updating the game state, and displaying the output"

A quick refresher – all MIDlets must contain the following methods:

```
public void startApp()
public void pauseApp()
public void destroyApp( boolean unconditional )
```

The WTK doesn't provide any skeleton code, so create MyGame.java in the src directory and enter the following skeleton (see Listing 1).

This code sets up a basic TextBox to be displayed on the screen, as in Figure 3, and ties an Exit button to the first available soft key on the device. On the default phone this is the top-left button. The commandAction method (from the CommandListener interface) ties the correct bit of functionality to the Command.

You'll notice that this MIDlet has an init() method that really doesn't do much. What's interesting is that this method is restricted so it can be run only once during the life of the MIDlet. Contrary to popular expectation, the startApp() method can in fact be called multiple times during the execution of a MIDlet. This isn't as important in the simple case we have so far, but will become necessary quite soon.

MIDP divides its UI library into two sets of classes, the *high-level API* and the *low-level API*. If you've played with MIDP before, then you're already familiar with the TextBox object and the other high-level API objects such as Alert, Form, and List. We don't want to spend any time discussing these here as they aren't very useful to game developers. We like to get our hands dirty and play with individual pixels, organizing the screen how we want it, so we need to use the low-level API. That's something we'll do right away by adding a splash screen to the MIDlet.

A Splash Screen

Splash screens are used at the start of a game. They're short animations or stills that are displayed for a few seconds and can usually be dismissed by the user at any time. It's a perfect opportunity to show off your team logo or a funky title screen for your game.

The following code implements a Splash class that's responsible for displaying and dismissing itself. The splash screen is dismissed either after a set period of time, or when the user presses any key or hits the touch-screen (if the device has one).

This allows us to add a splash screen to our code by just constructing a new instance of the Splash class. Technically, a Displayable cannot dismiss itself without having another Displayable to replace it, so the constructor takes two parameters: the Display and the Displayable that we want to follow our splash screen (see Listing 2).

The work of drawing anything to the screen is done by the paint method. In this case we just clear the screen and then draw the contents of a preloaded Image object into the graphics context. Note that we clear the screen first. In MIDP it's often beneficial to draw to the entire canvas. System screens that are thrown up by the device – perhaps an incoming telephone call – can interrupt the MIDlet screen. When those system screens are dismissed, the original MIDlet screen is left corrupted and needs to be fully redrawn. When the screen is interrupted in this way the canvas is informed by the reinvocation of its hideNotify() and showNotify() methods. The call to showNotify() is always followed by a call to paint().

The clear() method raises some notable points. First, we can see how to get some basic details about the device we're running on. The Display object exposes the size of the display and whether the device supports color or not – and if so, how many colors. We can also see that MIDP thinks there's only one active color and that all drawing commands will be rendered in this color. There's no Color class, so we must use a single integer to hold a grayscale shade (which can be used by

Goodbye Snake

An introduction to game development for MIDP handsets

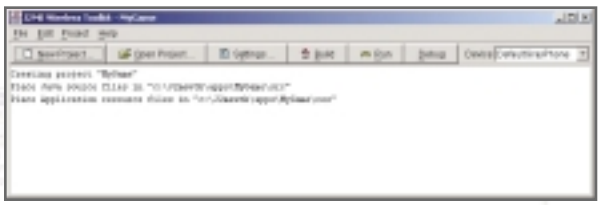


FIGURE 1 Creating the MIDlet project in KToolbar

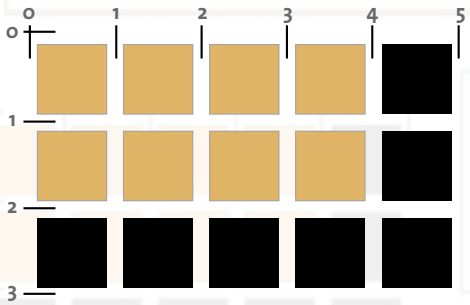


FIGURE 4 The MIDP pixel coordinate system

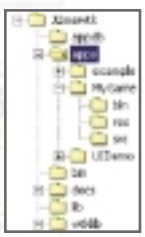


FIGURE 2 Directory structure

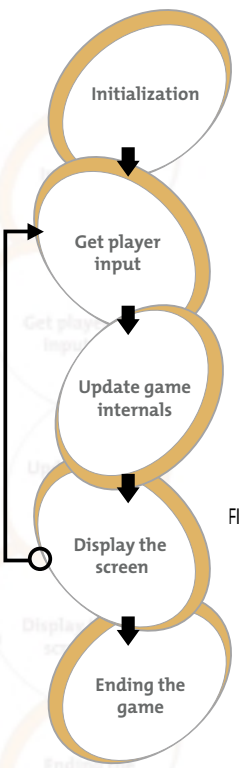


FIGURE 5 The game loop

FIGURE 3 Emulator screenshot

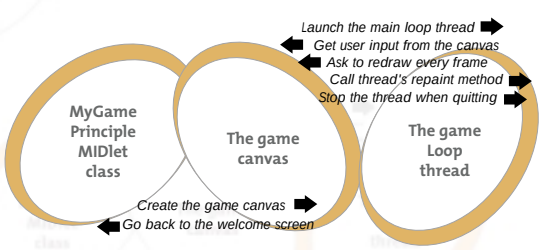


FIGURE 6 Architecture

a color device) or an integer RGB-triplet for colors.

The drawRect() and fillRect() methods are also worth mentioning. When drawing the outline of a rectangle (drawRect(...)), the outline is 1 pixel greater than the width and height specified. fillRect() works differently, however: fillRect(*x*,*y*,*w*,*h*) fills in the area inside the rectangle that would have been drawn by drawRect(*x*,*y*,*w*,*h*). In Figure 4, the black squares show the additional area that would be affected by the call drawRect(0, 0, 4, 2), compared to the purple squares, which show the rectangle created with fillRect(0, 0, 4, 2).

Basic Game Architecture

At the heart of every game is the main *game loop*, seen in Figure 5. Put simply, it's a basic series of procedures for getting the player's input, updating the game state, and displaying the output.

- ***Initialization:*** This is the stuff that happens at the start of the game: setting up the screen, showing intros, options menus, etc.
- ***Player input***: These are routines that take the player's input and store it in a way that allows the game state to change as necessary. We can handle this easily through the event-based API of the canvas using the following methods: keyPressed(), keyRepeated(), keyReleased(), pointerPressed(), pointerDragged(), pointerReleased(), and the CommandListener's commandAction() method.
- ***Update game internals***: This is the real meat of the game. Here we update all the game objects, check for collisions, update the scores, move the computer-controlled players, and so on.
- ***Display the screen***: Here, we convert the internal game state to the graphic that we then present on the screen. Each of the game's actors and background objects are drawn onto a graphics context. This can be done in two ways. Either the graphics can be drawn straight to the screen, or the graphics can be drawn to an off-screen buffer and when all the drawing is complete, this buffer is drawn onto the main screen in one go (also known as double buffering).

  Double buffering is an easy way to get flicker-free animation, because copying the entire off-screen buffer to the live display is usually a very fast process. Some MIDP devices support double buffering in hardware, so it's available transparently for free, programmatically speaking. You can test for this with the isDoubleBuffered() method of the canvas object.
- ***Ending the game:*** At the end of the game, there'll normally be some sort of ending sequence, perhaps an animation or high score table.

The Architecture

What we propose for a program architecture is explained in Figure 6.

A GameCanvas Skeleton

Your game canvas code can be structured roughly as shown in Listing 3. You can call the GameCanvas class from your main MIDlet class and then launch the main game loop thread from there, for example in a newGame() method.

Goodbye Snake

An introduction to game development for MIDP handsets

The paint() method needs to be called from your main game loop so that there's a screen redraw for every frame. You aren't supposed to call it directly. Instead the main game loop can utilize the repaint() method to request a repaint and the serviceRepaints() method to force waiting repaint requests to be executed immediately.

In the event-based methods keyPressed() and keyReleased(), you can test which keys have been pressed and set a state variable that will be read and processed by the game loop thread.

```
switch (getGameAction( param )) {
  case Canvas.LEFT:
       key_left = true;
       break;

  case Canvas.RIGHT:
       key_right = true;
       break;
  …
}
```

The game loop thread can be launched from the newGame() method by calling:

```
GameLoop gameLoop = new GameLoop( this );
Thread t = new Thread( gameLoop );
t.start();
```

A Main Game Loop Thread Skeleton

Listing 4 is an example of a main game loop thread skeleton. From here you can start developing some fun MIDlets, but be warned: you may discover there's a rough ride ahead. The current stock of MIDP devices aren't speed demons and the libraries are, arguably, still too small.

Author Bios

Mark Quinn is the games technical manager at the wireless entertainment company iFone.

Julien Tranchant is a developer at the games development house Aqua Pacific.

mark@ifone.com & jtranchant@nordnet.fr

Listing 1

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author mark
 * @version
 */
public class MyGame
  extends MIDlet
  implements CommandListener {

  private Display display;
  private boolean initialized = false;
  private Command exitCommand = new Command( "Exit", Command.EXIT,
1 );
  private Displayable dummy;

  /**
   *
   */
  public Ast() {
  }

  /**
   * Performs any initializations necessary.
   * Only runs once per MIDlet execution.
   */
  private void init() {
    if (!initialized) {
      display = Display.getDisplay( this );

      dummy = new TextBox( "Placeholder", "replace me with a game
canvas",
                           30, TextField.ANY );
      dummy.addCommand( exitCommand );
      dummy.setCommandListener( this );

      initialized = true;
    }
  }

  public void startApp() {
    init();
    display.setCurrent( dummy );
  }

  public void pauseApp() {
  }

  /**
   * Perform any cleanup and resource freeing necessary.
   */
  public void destroyApp( boolean unconditional ) {
    // we don't need to do anything here yet
  }

  /**
   * Respond to commands, including exit
   * On the exit command, cleanup and notify the execution envi-
ronment
   * that the MIDlet has destroyed itself.
   */
  public void commandAction(Command c, Displayable s) {

    if (c == exitCommand) {
      destroyApp( false );
      notifyDestroyed();
    }
  }
}
```

Listing 2

```
import java.util.*;
import javax.microedition.lcdui.*;

public class Splash
  extends Canvas {

  private Display display;
  private Displayable next;
  private Timer timer = new Timer();

  public Splash( Display display, Displayable next ) {
    this.display = display;
    this.next = next;

    display.setCurrent( this );
  }

  protected void showNotify() {
    timer.schedule( new TimerTask() { public void run() {
displayNext(); }}
                    , 10000 );
  }

  protected void hideNotify() {
    timer.cancel();
  }


  protected void keyPressed( int keyCode ) {
    displayNext();
  }

  protected void pointerPressed( int x, int y ) {
    displayNext();
  }

  private void displayNext() {
    display.setCurrent( next );
  }
```


```
  private void clear( Graphics gfx ) {

    int r=0, g=0, b=0, gray=0;
    boolean color = display.isColor();

    // save out the current color
    if (color) {
      r = gfx.getRedComponent();
      g = gfx.getBlueComponent();
      b = gfx.getGreenComponent();
    }
    else {
      gray = gfx.getGrayScale();
    }

    // clear the graphics context
    gfx.setGrayScale( 255 );
    gfx.fillRect( -1, -1, getWidth() + 1, getHeight() + 1 );

    // restore the current color;
    if (color) {
      gfx.setColor( r, g, b );
    }
    else {
      gfx.setGrayScale( gray );
    }
  }

  protected void paint( Graphics g ) {
    clear( g );

    Image logo = null;
    try {
      logo = Image.createImage( "/images/logo.png" );
    }
    catch (java.io.IOException e) {}

    g.drawImage(logo, 0, 0, g.LEFT|g.TOP);
  }

}
```

Make the following additions to MyGame.java.

```
  /**
   * Performs any initializations necessary and displays the
splash
   */
  private void init() {
    display = Display.getDisplay( this );

    dummy = new TextBox( "Placeholder", "replace me with a game
canvas",
                         30, TextField.ANY );
    dummy.addCommand( exitCommand );
    dummy.setCommandListener( this );

    showSplash( display, dummy );

    initialized = true;
  }


  public void showSplash( Display d, Displayable next ) {
    new Splash( display, next );
  }
```

Listing 3

```
import javax.microedition.lcdui.*;

public class GameCanvas
  extends Canvas
  implements CommandListener {

  /**
   * Creates new GameCanvas
   */
  public GameCanvas( MyGame myGame ) {
    // Do initialization and set an exit soft button

    addCommand( new Command( "Exit", Command.EXIT, 0 ) );
    setCommandListener(this);
  }

  public void paint( Graphics g ) {
    // Paint everything for the game
    // First, erase the screen and then display the background and
actors
  }

  public void newGame() {
    // Start a thread which contains a game loop
  }

  private void quitCanvas() {
    // stop the realtime thread
  }

  public void quit() {
    // return to the welcome screen
  }


  protected void keyReleased( int param ) {
    // process keys when released
  }

  protected void keyPressed( int param ) {
    // process keys when pressed
  }

  public void commandAction(Command p1, Displayable p2) {
    if (p1.getCommandType() == Command.EXIT)
      quitCanvas();
  }
}
```

Listing 4

```
import javax.microedition.lcdui.*;
import java.util.*;

public class GameLoop
  implements Runnable {

  private GameCanvas gameCanvas;

  public GameLoop( GameCanvas gameCanvas ){
  }

  private void init(){
  }

  private void processKeys(){
  }

  private void moveObjects(){
  }

  private void processCollision(){
  }

  // Thread entry when started
  public void run() {

    while(gameLoopState == STATE_RUN) {

      processKeys();

      moveObjects();

      processCollision();

      // force the game canvas to paint the frame
      gameCanvas.repaint();
      gaemCanvas.serviceRepaints ();
    }
  }

  // the function called by the game canvas every frame which
draws graphics
  public void paint( Graphics g ) {

  }

  // quit the thread loop
  public void quit() {

  }

}
```

KBrowser

by 4thpass, Inc.

REVIEWED BY JASON BRIGGS
jasonbriggs@sys-con.com

Before we get started, it's worthwhile noting that I'm not a big fan of WAP. Well, not the protocol itself exactly, but WML-based Web sites – the whole idea of browsing the Web on your mobile seems somewhat flawed when your screen real estate is minimal and you probably have only one more color than Henry Ford wanted for his cars.

So when 4thpass offered me a look at their KBrowser for MIDP and Palm devices, I wasn't expecting to be that enthused. Sure, it's written in Java – feel free to clock me over the back of the head with a large shoe if I ever turn up my nose at a look at a good Java application – but it's WAP. If you've ever used a WAP phone (or know someone who has) you undoubtedly know the rhyme that goes along with it: WAP is crap. Okay, so maybe you haven't heard it, and I hang out with some really unimaginative people.

But I have to say, this is one neat little product. The evaluation version 4thpass sent to me was an enormous 69KB in size and easy to install on my Zucotto WHITEboard device emulator; soon I was browsing WAP sites to my heart's content (slight exaggeration there, but you get what I mean). The KBrowser supports various industry standards, including WAP 1.2, HTTP, WMLScript, WTLS Security, and the draft WAP 2.0 – not bad going for an application that's smaller than a lot of normal Web-site graphics.

A Change of Mind

In the end I actually enjoyed checking out a few WAP sites. The TV Guide on Yahoo seems like something I would use fairly regularly, since I'm consistently forgetting to buy the paper on Thursday (the free weekly TV guide day), and have to turn on my computer just to check out what's on television.

Who needs a browser built into the phone when you can just download the KBrowser onto your J2ME-capable device!

Figure 1 provides some shots of KBrowser in action. Clockwise from top left: an image alignment test in the KBrowser Test Suite; the weather forecast from the test suite; ; the menu of the "Kung fu training" game at www.kung-fuboy.com/wap/; an "almost-subliminal" advert on the Docklands Light Rail travel information site (www.kizoom.co.uk/dlr/i.jsp); the cartoon site – wap.cartoonscape.com; the Yahoo UK TV Guide.

FIGURE 1 KBrowser in action

Technical Information

4thpass provided some background technical information on its KBrowser development efforts, and Javed Chaudry, director of marketing/communications, was kind enough to answer my questions.

<briggs>: **How long did the browser take to develop?**

<chaudry>: We began with a KBrowser Palm edition and leveraged much of that expertise into the J2ME version, so the overall process was about two years.

<briggs>: **How big is/was the team?**

<chaudry>: This was a collaborative effort for 4thpass, Inc.

<briggs>: **Why develop a browser? How big do you expect your market to be – and what is your intended market for that matter?**

<chaudry>: The KBrowser proved that Java and WAP don't have to be competing technologies. Java can leverage WAP for its primary use, the transport. The KBrowser also completes our end-to-end provisioning solution (the mobile application system). Our provisioning solution works with *any* J2ME-enabled device, but we offer the KBrowser as an optional component.

We're working with companies like Sun, RIM, and LG TeleCom in Korea to name a few. We expect the market to be very large. As the number of J2ME devices grows, so grows the market. The Java marketplace now has buy-ins from companies such as Motorola, Nextel, Nokia, Siemens, and Sprint.

<briggs>: **Which parts of the application were easier to develop because of Java?**

<chaudry>: No parts in particular stand out. Java's lack of pointers and use of garbage collection tend to help Java development in general.

<briggs>: **Do you have any recommendations or advice for other J2ME developers?**

<chaudry>: We would tell developers to remember three things: code size is king, speed is queen, and heap size is an unfriendly concubine. They all fight, constantly.

Editor's Note

This was the first in a series of articles on software developed for the J2ME platform. If you have developed a J2ME app (MIDlet or PersonalJava) that you think ***JDJ*** should feature, let us know.

info:

4thpass, Inc.
83 South King Street, Suite 100
Seattle, WA 98104
Web: www.4thpass.com
Phone: 206 749-9070
Fax: 206 749-9036
E-mail: sales@4thpass.com

Specifications:
Operating software: J2ME 1.0 or higher, MIDP 1.0 or higher
Memory requirements: 128K runtime memory; 65K flash memory without WMLScript or WAP Stack; 115K flash memory including WMLScript without WAP Stack; 170K flash memory including WMLScript and WAP Stack

EXTREME PERFORMANCE TUNING

PART 3 of 3

Creating graphical programs that use Swing or AWT

Written by James McGovern

Many development shops have used J2EE to build a successful business-logic tier but have fallen short on obtaining the desired look and feel. On my current project we considered using applets as substitutes for GIF-based buttons, creating a utility to modify tree-based structure data as well as an application that will allow a secure file-based transmission. In my spare time I'm also working on an idea for a video game. All of these require an architecture that takes performance into consideration.

Performance tuning is an integral part of any Java-based application development effort. It's vitally important for the users of your program to think that it performs well. In Parts 1 and 2 (***JDJ***, Vol. 6, issues 9 and 10), we focused on tips that are geared toward making J2EE-based applications run faster and scale better. Part 3 focuses on making programs that are graphical in nature (applets and applications) and utilize either Swing or AWT to become extremely scalable.

Always JAR Classes

A Java Archive (JAR) is a file that has been compressed according to the JavaBeans standard. It's the primary and recommended method for delivering JavaBean components. It helps reduce file sizes and download times, which may make your applets appear faster. A JAR file can contain one or more related beans, support files such as graphics, sounds, HTML, and other resources.

To specify a JAR file within an HTML/JSP file, add the variable ARCHIVE = "name.jar" to the applet tags.

Consider Using Delegated Loading

Have you ever visited a Web site that uses Java applets and noticed that your browser puts a placeholder where the Java applet is supposed to run? What happens when the applet takes a significant time to load? The most likely answer is that the user goes to another site. In this scenario it would be useful to display a message that the applet is being loaded and he or she should stick around.

Let's discuss one technique that can help you realize this goal. First you need to create a very small applet that downloads the real applet in the background. Let's stub out some code snippets that show how this can be accomplished (see Listing 1).

The compiled code should be less than 2K so it downloads quickly. In looking at this code in Listing 1 you'll notice a couple of subtle items. First we implement AppletStub. Typically an applet determines its code base from the caller. In our scenario we have to tell the applet where to retrieve this information by calling setStub. The other notable difference is that the AppletStub interface implements many of the same methods in the Applet class, except for AppletResize. We're delegating the AppletResize method to the resize method.

Always Set the Display Mode

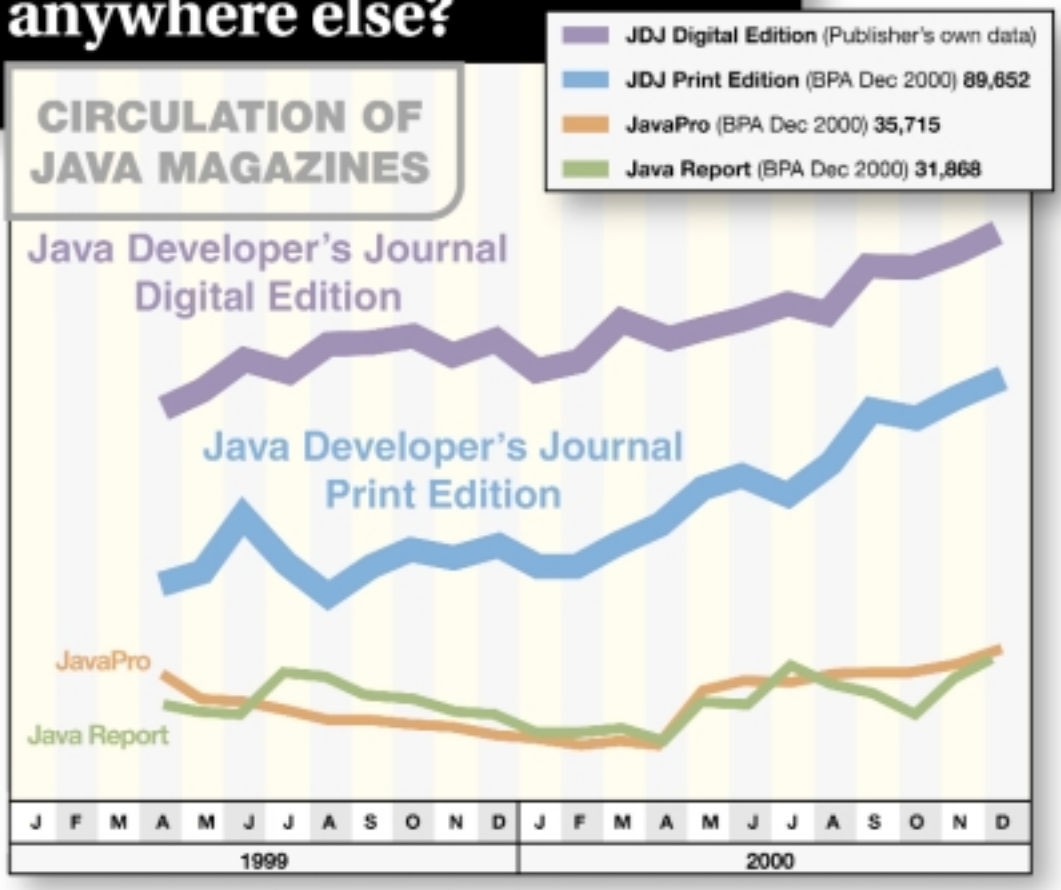
Setting the display mode within an AWT-based application allows your program to run quicker if the images it chooses to display share the same bit-depth as the screen. An added advantage of setting the display mode comes when you know the display size in advance, as you won't have to programmatically scale images up or down depending on the display. *Note:* In some environments you can only change the display mode when in full-screen exclusive mode.

You can determine the current display mode by calling java.awt.getDisplayMode() as well as a list of all supported modes by calling java.awt.getDisplayModes(). To set the display mode you need to pass the width and height of the monitor in pixels, bit depth (number of bits per pixel), and refresh rate (how often the monitor updates itself). Listing 2 demonstrates setting the display mode.

Full-screen exclusive mode is handled through a java.awt.GraphicsDevice object. This particular object

It's a no-brainer

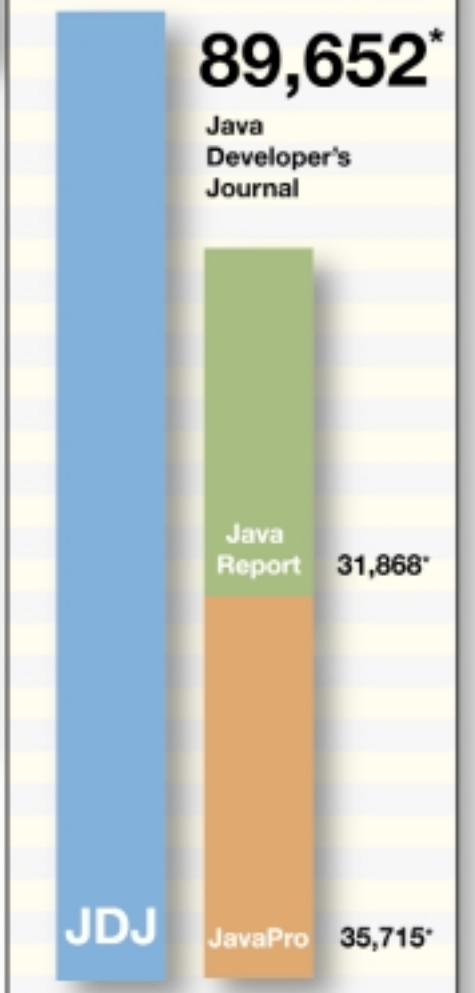
Why would you advertise anywhere else?



Your ad in *JDJ* reaches twice as many recipients as all other Java pubs combined for 1/2 the cost...*

*Independent survey

They Don't Quite Stack Up



JDJ reaches more Java developers than all other Java magazines COMBINED!

* Most recent six month average monthly circulation of three magazines as per BPA International

SYS-CON MEDIA

www.JavaDevelopersJournal.com

(201) 802-3020

exposed another useful method, isFullScreenSupported(), which determines if full-screen exclusive mode is available. It's possible to still set full-screen mode on a system that doesn't support it. The outcome may be performance degradation. In this scenario it may be better to run the application in a windowed mode with a fixed size rather than setting a full-screen window.

Consider Preloading Images Before Painting

The ImageObserver interface can be used to receive notification when an image is being loaded. The ImageObserver interface has only one method, imageUpdate(), which can be used to paint the image on the screen in a single repaint. Listing 3 provides a technique for doing this.

The ImageUpdate method is called when information about an image becomes available. This method returns true when further updates are needed and false if the required information has been obtained.

Always Override Update

The default behavior of update is to clear the screen of all content and call the paint() method. Applications that are heavy in graphics will display a flickering behavior when using the default behavior.

To prevent the clearing of the screen before paint is called, you can override update() simply as follows:

```
Public void update(Graphics g) {
    paint(g);
}
```

An even better method would be to override the update() method and paint only the region of the screen where changes will take place. Here's a better example:

```
Public void update(Graphics g) {
    g.clipRect(x, y, w, h);
    paint(g);
}
```

> **The main reason for performance degradation in a graphical-based application can be traced back to inefficient redrawing**

Consider Delaying Redrawing

The main reason for performance degradation in a graphical-based application can be traced back to inefficient redrawing. This is usually noticed when a user resizes and/or scrolls a window. This behavior causes redraw events to be generated faster than the redraw can execute. The best way to handle this deficiency is to ignore all events that arrive too late.

One technique that can help is to draw to a large off-screen buffer (explained later). The redraw event listener can copy parts of a bitmap. This technique will work in some situations, but wouldn't be useful in any situation where image dimensions actually change, such as 3D rotation. Another technique may be to have the redraw event listener block further redraws until the current one finishes. We could essentially take the last received redraw event and discard all others.

I recommend introducing a couple of milliseconds into the equation so that if we immediately receive another redraw event, we can stop our current event and process the last redraw received; otherwise, we continue with the current redraw.

Listing 4 demonstrates a simple technique for handling events but could be expanded to handle worker threads. It's usually a good idea to spawn a worker thread whenever an event initiates time-consuming tasks. Otherwise, all components will freeze as only one event can occur at a time.

Always Double Buffer

Draw your images off screen and then display an entire image at once. There are two buffers and you can switch between them. This allows you to offload the drawing of the display image to the background using a lower-priority thread, making your program use unused clock cycles. Here's a quick code snippet that displays the technique:

```
Graphics graphics;
Image offscreenImage = createImage(size().width, size().height);
Graphics offscreenGraphics = offscreenImage.getGraphics();

offscreenGraphics.drawImage(img, 50, 50, this);
graphics.drawImage(offscreenImage, 0, 0, this);
```

Use BufferedImage

Java JDK 1.2 uses a software renderer so that text appears similar across platforms. The implementation of this function requires direct access to the pixels that form the text. Prior JDKs had performance issues with this technique as it involved copying lots of bits around in memory. The Java specification to handle this particular performance problem implemented a new image type, BufferedImage.

> **The ability to offload work and have** parallel streams of work happening concurrently frees up the CPU and systems bus

The BufferedImage subclass describes an image with an accessible buffer of image data. A BufferedImage is comprised of a ColorModel and a raster of image data. This class typically uses the RGB (red, green, blue) color model but can also handle grayscale. The constructor is straightforward and looks like:

```
Public BufferedImage (int width, int height, int imageType)
```

ImageType allows us to specify the actual type of image we want to buffer such as 5-bit RGB, 8-bit RGB, grayscale, and so forth.

Consider Using VolatileImage

Many hardware platforms and their respective operating systems support basic hardware acceleration. Hardware acceleration typically provides rectangle filling that's faster than having the CPU perform the function in an offloaded fashion. The ability to offload work and have parallel streams of work happening concurrently frees up the CPU and systems bus. This also allows for nongraphics related functionality to occur, making the application faster.

VolatileImage allows applications to create hardware-accelerated, off-screen images as well as manage its contents. Since this feature takes advantage of the underlying platform's capabilities, performance increases depend primarily on the graphics adapter in use.

VolatileImages can be lost at any time, hence they're volatile; it's a good idea to check for loss of content before using the image. The VolatileImage has two methods to test for content loss:

Missed an issue?

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

JAVA DEVELOPER'S JOURNAL

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

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

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

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

XML JOURNAL

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

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

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

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

COLDFUSION Developer's Journal

The most complete library of exclusive *CFDJ* articles!

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

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

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

CFDJ The Complete Works
~~Reg $79.99~~
Buy Online
Only $71.99

CF Advisor

The most complete library of exclusive *CFA* articles!

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

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

CFA The Complete Works
~~Reg $79.99~~
Buy Online
Only $71.99

JDJStore.com

Order Online and Save 10% or More!

www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

```
Public abstract int validate(GraphicsConfiguration gc);
Public abstract Boolean contentsLost();
```

The validate() method should be called whenever copying from or rendering to a VolatileImage object. The contentsLost() method tells the program if the contents of the image have been lost since the last call to validate().

Although VolatileImage is an abstract class, don't subclass from it. VolatileImage should always be created by calling Component.createVolatileImage().

Consider Using Window Blitting

Typically when a user scrolls, the entire visible contents are redrawn. This may cause a lot of redrawing work for no good reason. Window blitting is used by many graphical OS subsystems, including the WIN32 GDI, MacOS, and X/Windows. It moves bits directly from the screen buffer to their new location when scrolling. This technique updates only the newly exposed area. To enable window blitting for your Swing-based application, use the following method:

```
setScrollMode(int mode);
```

Using this technique will increase scrolling performance in most applications. The only type of application that I'm aware of where this technique will cause a performance decrease is when the scrolling application is being scrolled in the background. If the user is scrolling the application, it will always be in the foreground and you don't have anything to worry about.

Conclusion

I hope this article has been useful and you get the opportunity to try some of the recommended techniques.

Author Bio

James McGovern is an enterprise architect with Hartford Technology Services Company, L.L.C., an information technology services firm dedicated to helping businesses gain competitive advantage through the use of technology. His focus is on developing extremely scalable Internet applications.

james.mcgovern@thehartford.com

Listing 1

```
import java.applet.Applet;
import java.applet.AppletStub;
import java.awt.Label;
import java.awt.Graphics;
import java.awt.GridLayout;

public class PreLoader extends Applet implements Runnable, AppletStub {
    String appletToLoad;
    Label label;
    Thread appletThread;

    public void init() {
        // This is the applet we want to delay loading
        appletToLoad = getParameter("applet");
        label = new Label("Stay tuned" + appletToLoad);
        add(label);
    }

    public void run(){
        try {
            // Get the class of the real applet
            Class appletClass = Class.forName(appletToLoad);

            // Create an instance of the real applet
            Applet realApplet = (Applet)appletClass.newInstance();

            // Set the stub for the applet
            realApplet.setStub(this);

            // Get rid of the Stay Tuned message
            remove(label);
```

2002

JUNE 24-27

JACOB JAVITS CONVENTION CENTER

NEW YORK, NY

JAVA, XML AND .NET TECHNOLOGIES

NJ

James Gosling,
The Father of Java

web services conference&expo EDGE™

INTERNATIONAL WEB SERVICES CONFERENCE & EXPO

JDJ conference&expo EDGE™

INTERNATIONAL JAVA DEVELOPER CONFERENCE & EXPO

XML conference&expo EDGE™ 2001

INTERNATIONAL XML CONFERENCE & EXPO

Fundamentally Improving the Speed,
Cost & Flexibility of Business Applications

Who Should Exhibit...

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

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

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

Who Should Attend...

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

THE CALL FOR PAPERS

WILL OPEN ON NOV. 30, 2001

VISIT WWW.SYS-CON.COM FOR DETAILS

FOR MORE INFORMATION

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

201-802-3069

WWW.SYS-CON.COM

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

```
            // Maximize the applet to the current size
            setLayout(new GridLayout(1, 0));
            add(realApplet);

            // Lets now display the real applet
            realApplet.init();
            realApplet.start();
        }
        catch (Exception ex) {
            // Display an error
            label.setText("Error loading applet.");
        }

        // We need to redraw the screen
        validate();
    }

    public void appletResize(int width, int height) {
        // Transfer the call of AppletResize from the stub to the
applet.
        resize(width, height);
    }
}
```

Listing 2

```
GraphicsDevice aDevice;
Windows aWindow;
DisplayMode newDisplayMode;
DisplayMode oldDisplayMode = aDevice.getDisplayMode();

try {
    aDevice.setFullScreenWindow(aWindow);
    aDevice.setDisplayMode(newDisplayMode);
    // do rest of work
}
finally {
    aDevice.setDisplayMode(oldDisplayMode);
    aDevice.setFullscreenWindow(null);
```

Listing 3

```
public boolean imageUpdate(Image img, int flags, int x, int y, int w, int
h) {
    if ((flags & ALLBITS) !=0 {
        repaint();
    }
    else if (flags & (ABORT | ERROR)) != 0) {
        error = true;
        // File was not found so consider displaying placeholder
        repaint();
    }
    return (flags & (ALLBITS | ABORT | ERROR)) == 0;
}
```

Listing 4

```
public static void runOnce(String id, final long milliseconds) {
        synchronized(s_queue) { // s_queue is a collection of all events
            if (!s_queue.containsKey(id)) {
                s_queue.put(token, new Latest());
            }
        }

        final Latest latest = (Latest) s_queue.get(token);
        final long time = System.currentTimeMillis(); // Get current time
        lastest.time = time;

        (new Thread() {public void run() {
            if (milliseconds > 0) {
                try {Thread.sleep(milliseconds);} // Sleep for a little
bit
                catch (Exception ex) {}
            }
            synchronized(lastest.running) { // Wait until previous fin-
ishes
                if (latest.time != time)  // Only run latest
                return;
            }
        }}).start();
}

private static Hashtable s_queue = new Hashtable();

private static class Latest {
    public long time=0;
    public Object running = new Object();
}
```

Download the Code! www.JavaDevelopersJournal.com

wireless EDGE conference&expo

INTERNATIONAL WIRELESS BUSINESS & TECHNOLOGY CONFERENCE & EXPO

CONFERENCE & EXPO

CONFERENCE & EXPO

Plan to Exhibit

Provide the Resources to Implement Wireless Strategy

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

Shaping Wireless Strategy for the Enterprise

Santa Clara, CA

May 7-9, 2002

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

Plan to Attend the 3-Day Conference

WHO SHOULD ATTEND

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

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

Conference Tracks

Track One: Development	Track Two: Connectivity	Track Three: Wireless Apps	Track Four: Hardware	Track Five: Business Futures
WAP	Smart Cards	Education	Cell Phones/ WorldPhones	Wireless in Vertical Industries
i-Mode	Wireless LANs incl. Bluetooth	Health Care	PDAs	The WWWW
Bluetooth / 802.11	UMTS/3G Networks	Entertainment	Headphones/ Keyboards / Peripherals	Unwired Management
Short Messaging	Satellite Broadband	Transport	Transmitters Base Stations	From 3W to 4W: Issues and Trends
Interactive Gaming		Financial Services	Tablets	"Always-On" Management
GPS / Location-Based		Supply Chain Management		Exploiting the Bandwidth Edge
Wireless Java				Unplugged Valueware
XML & Wireless Technologies				Wireless Sales & Marketing

FOR INFORMATION CALL

201 802-3084

SPEAKER PROPOSALS INVITED

WWW.SYS-CON.COM

SHAPE YOUR WIRELESS STRATEGY! SAVE THE DATES!

EXCLUSIVE SPONSORSHIPS AVAILABLE

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

SYS-CON MEDIA

SYS-CON EVENTS

What's Online...

November **2001**

JDJ Online

Check in every day for up-to-the-minute news, events, and developments in the Java industry. Can't get to the newsstand on time? Visit www.javadevelopersjournal.com and be the first to know what's happening in the industry.

JDJ Readers' Choice Awards

See how your favorite products fared in this year's ***JDJ*** Readers' Choice Awards. A complete list of winners and finalists is here, along with product descriptions, voting percentages, and more.

JDJEdge Highlights

On September 23–26, the Hilton New York was filled with JDJEdge Conference & Expo attendees for the East Coast's largest Java developer conference. All the highlights of the show are here – James Gosling's insightful keynote, the Web services panel discussion, pictures, and **SYS-CON Radio** interviews with all the industry heavyweights.

Save the date: next year's conference will be at New York's Jacob Javits Convention Center, June 24–27, 2002.

Reader Feedback

Read Scott McNealy's editorial, "The Platform of Choice," written exclusively for ***JDJ***, and tell us what you think. The following are excerpts from what our readers had to say:

Support the platform that runs on multiple operating systems. Java is the right choice.

—Amadou Amadou

You guys are missing the point. As long as Microsoft controls 90% of the client desktops in the world, they can do whatever they want with Java or any other program or service that is accessed through a Windows client. The government won't stop them; only the consumer can.

—Mark Ashworth

Nice words, Mr. McNealy. When will you let go of Java? Everything you said will make sense when you release Java to the "open source" community.

—Unmesh Malvankar

Precisely! The platform of choice is certainly Java for all the reasons Scott and others have stated. But for how long?

—Karun

JDJ Store CD Special

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

Order now and have over 1,000 articles on hand for research and review. There are features, how-tos, product reviews, case studies, tips and tricks, interviews, IMHOs, and more!

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

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

JDJ NEWS

Idetix Introduces Revize 3.5

(*Troy, MI*) – Idetix Software Systems Inc. announces the availability of the Revize v3.5 Content Management and Delivery System (CMDS). A platform and Web server–independent content management software built in pure Java, Revize enables nontechnical content editors to manage and change their own content and workflow permissions in a "management"-controlled environment.

www.idetix.com

ATG Announces New Enterprise Commerce Applications

(*Cambridge, MA*) – Art Technology Group, Inc., announced the ATG Enterprise Commerce Suite, which includes the ATG Dynamo e-Business Platform, ATG Dynamo Application Server, ATG Control Center, and rich B2B and B2C commerce applications for the enterprise market.

For enterprises looking only at B2C retail commerce capabilities, ATG also announces the ATG Consumer Commerce Suite.

Unique to the platform are deep, scenario-driven personalization capabilities that help organizations customize online interactions and automate routine business commerce processes.

www.atg.com

Centrifusion and iPlanet Announce Partnership

(*Chicago*) – Centrifusion, Inc., an e-business consulting and integration firm, and iPlanet E-Commerce Solutions, a Sun-Netscape Alliance, announced the formation of a formal alliance to build, market, and service e-commerce infrastructure solutions.

Both Centrifusion and iPlanet serve Global 3500 industry leaders in a variety of fields, including finance, telecommunications, and media.

www.centrifusion.com

Kada to Bring Java to the Desktop Phone

(*Burlington, MA*) – Cisco Systems Inc. has selected the Kada Mobile Platform to support Java application capabilities in Cisco's family of Voice over Internet Protocol (VoIP) phones.

The Kada Mobile Platform for Java delivers a full set of tools that enable organizations to implement Java solutions that are compliant with the J2ME specifications and to build adjunct profiles specific to their application needs while maintaining compliance with the specifications.

www.kadasystems.com

TogetherSoft Launches Together ControlCenter 5.5

(*Raleigh, NC*) – TogetherSoft Corporation's Together ControlCenter 5.5 release enables companies to extend their current technology infrastructure and provide their development teams with a platform for modeling, building, and deploying software solutions.

Key enhancements include support for VB 6, VB .NET and C# (early access), Oracle9*i* and DB2 7.1 database management systems, and J2EE-compatible and SOAP/Web services support.

www.togethersoft.com

THOUGHT Inc. Opens EU Headquarters in Paris

(*San Francisco*) – THOUGHT Inc., a leader in object to relational (O/R) mapping technology, OPTIMIZED for EJB and Java, announced the official opening of its EU operations headquarters in Paris. The offices at 71 rue de Saussure F-75017, Paris, are located in a high technology neighborhood similar to San Francisco's "South of Market."

www.thoughtinc.com

SYS-CON Events Moving to Javits Center in 2002

(*Montvale, NJ*) – **SYS-CON Events, Inc.**, will relocate its East Coast events to the Jacob Javits Convention Center in New York City in 2002. Web Services Edge 2002 East–International Web Services Conference & Expo and JDJEdge 2002–International Java Developer Conference & Expo will take place June 24–27, 2002. **SYS-CON**'s East Coast events will complement the Technology Exchange Week New York – TECHXNY – a weeklong showcase for the most dynamic business technology players and products in the world.

"We look forward to coming back to New York City for the third year in a row and plan to double our exhibit floor by moving to the Jacob Javits Convention Center," said Grisha Davida, vice president of business development at **SYS-CON Media**. "This will allow us to host Java, XML, .NET, and all related technologies under our Web services canopy."

"We will open the Call for Papers for next year's conference on October 22, 2001," stated Cathy Walters, vice president of **SYS-CON Events**.

www.sys-con.com

SYS-CON Media Launches *Web Services Journal*

(*Montvale, NJ*) – **SYS-CON Media** has debuted its premier issue of ***Web Services Journal***, the first print and online magazine devoted exclusively to the newest and most pervasive computing paradigm since the arrival of the Web itself.

The premier issue features articles from some of the best-informed developers, analysts, and executives in the *i*-technology world. This collector's issue includes a "CEO Round Table Discussion" led by executives and visionaries like David Litwack, CEO of SilverStream; Barry Morris, CEO of Iona, David Clarke, CEO of Cape Clear; Simon Phipps of Sun Microsystems; and Tyler Jewell of BEA Systems, among others.

www.sys-con.com/webservices

SYS-CON Media, Group Intelligence in Strategic Alliance

(*Montvale, NJ*) – **SYS-CON Media** and Group Intelligence have announced a strategic alliance in support of the fastest-growing Web-infrastructure market, WebSphere.

The companies will jointly deliver a full range of print, event, online, and digital services, creating the first information resource and infrastructure (InfoStructure) of the WebSphere marketplace.

WebSphere Developer's Journal, the newest *i*-technology publication from **SYS-CON Media**, will be published monthly starting in November.

www.sys-con.com
www.groupintelligence.com

JDJ Announces Record Circulation More Than 136,000 Copies Distributed

(*Montvale, NJ*) – **SYS-CON Media** has announced that circulation of ***Java Developer's Journal*** reached 104,254 copies with the June 2001 print edition, generating single-issue revenues of over $1 million. ***JDJ*** also announced a six-month target circulation of 97,325 (up from the previous six-month average of 89,652) during the period ending June 2001. In addition, ***JDJ*** delivered a 236,715-copy monthly circulation of its digital edition with the September 2001 issue.

"We're very pleased to see that ***Java Developer's Journal*** is serving the fast-growing Java developer community around the globe as the hands-down leading source of print and online information, and its lead increases with every issue," reports Fuat Kircaali, founder and CEO of **SYS-CON Media**. "Our hard-working **SYS-CON** team is very happy to be part of a leading company that offers unmatched quality information products in print and online, and with our *i*-technology conferences."

www.sys-con.com

Next Month

J2EE CTS: WHAT'S IT TO YOU?
An overview of the CTS for J2EE component developers
by Jon Maron

WABA ON WHEELS
A Java-powered Palm Pilot robot
by James Caple

PRODUCT REVIEW
WHITEboard SDK, Bluetooth Edition by Zucotto Wireless Inc.
reviewed by Jason Briggs

J2ME BENCHMARKING: A REVIEW
Evaluating an application's performance objectively
by Carl Barratt and Glenn Coates

JAVA JOBS
A Year of Advice
The year in review
by Bill Baloglu and Bill Palmieri

JAVA DEVELOPER'S JOURNAL

Just Don't Do It

How to blow a great opportunity

WRITTEN BY
BILL BALOGLU & BILLY PALMIERI

We have many stories of skilled engineers who've done a great job on contract and full-time positions, leaving behind happy managers who sing the engineer's praises for the rest of their days.

We've also seen engineers self-destruct at each step of the hiring process, eliminating themselves from consideration for what could have been great opportunities.

As technical hiring has shifted from employees to employers calling the shots, it's especially important to pay heed to the errors of those who have come (and gone) before.

How to Blow an Interview

- ***Don't pay attention:*** Look around the room, show more interest in photos on the wall and books in the bookcase than in talking to the interviewer. Okay, so not everyone is comfortable in a face-to-face interview. But if you want to make a good or even decent impression, force yourself to focus on the interviewer and respond directly with pertinent answers.
- ***Challenge and contradict the interviewer:*** You know more about technology than the hiring manager, right? Maybe you do, maybe you don't. But just to show them how smart you are, after they explain the project tell them it won't work. Tell them they're making a big mistake and going about it all wrong. You'll make their job a lot easier (instantly reducing their list of candidates by one).

  If you do have another idea of how to execute the type of project they've described, after listening carefully to the project description (and convincing them that you could do it), phrase other suggestions with, "Have you considered using...?" This shows your respect for their current plan, and your willingness to offer meaningful input.
- ***Look like something the cat dragged in:*** No matter how brilliant you are, a dirty t-shirt, unkempt hair, or torn jeans are universal turnoffs. Believe me, we've seen it all. If you smoke, make sure your clothes don't reek, don't light up right before the interview, and pop a few breath mints before going in.

  You may be as casual as you like on the job, but for an interview, clean up your act. It shows respect for yourself and for the people who are interviewing you. It also communicates a professional image that many engineers don't think to convey.
- ***Put the interviewer on the defensive:*** The point of an interview is to make sure the interviewers are who they says they are, right? Grill the interviewer, cross-examine her or him, then get up and leave because you just blew the interview.

  The point of an interview is to learn about the project, answer questions about your skills and expertise, and convince the interviewer that you're the best person for the job. Intelligent questions about the project and the company go a long way, but remember that you're there to sell yourself.
- ***If it's not your dream job, tell them you're not interested:*** Maybe the position described isn't your ideal first choice, maybe you've got a few other irons in the fire. But unless it's a serious mismatch of skills (like you're a Java architect and they're looking for a DBA), don't walk away from an interview leaving the impression that you don't want the job.

  Leave the manager with the option of offering you the job. In today's tight market it's especially important to keep all of your options open. First choices often fall through – and you can always politely pass on the offer later. This leaves the door open for them to consider you down the road for what might be an even better opportunity.
- ***Never mind about this job, focus on the next one:*** Many positions, both contract and full-time, have the potential to lead to bigger and better things, but that's after you've proven yourself to be a valuable asset to the company.

  A first interview is not the time to focus too much on promotions and the potential for advancement. It can give the impression that you're only looking at their current project as a foot in the door. Get yourself in, do a great job on the project, and then talk about where you might go from here.

How to Self-Destruct on the Job

Many things can go wrong on any given project. But some of the people we've crossed paths with seem to have gone out of their way to get themselves fired.

- ***Catch some z's:*** Sleeping on the job has been a timeless method of angering employers ever since the first caveman fell asleep on the night watch and let the village fire go out. And it's still a surefire way to get canned today. Would you pay us to sleep?
- ***Personal perspective:*** Have you heard the one about the contract consultant working at a major Silicon Valley firm who started an e-mail war with a co-worker from a different cultural background? It's true. It happened. He was fired.
- ***Fight the power:*** For the quickest, most effective way to implode on the job, nothing beats picking a fight with your manager. Get into heated arguments in front of other members of the group, and try to rally co-workers in your battle against your manager.

  And if you've still got a job, try getting your manager in trouble with a superior and leave threatening messages at the manager's home number. What's a little restraining order between friends? ✎

billb@objectfocus.com

billy@objectfocus.com

AUTHOR BIOS

Bill Baloglu is a principal at ObjectFocus (www.ObjectFocus.com), a Java staffing firm in Silicon Valley. Previously he was a software engineer for 16 years. Bill has extensive OO experience and has held software development and senior technical management positions at several Silicon Valley firms.

Billy Palmieri is a seasoned staffing industry executive and a principal at ObjectFocus. His prior position was at Renaissance Worldwide, a multimillion-dollar global IT consulting firm, where he held several senior management positions in the firm's Silicon Valley operations.

Our New World

WRITTEN BY
BLAIR WYMAN

Our world has been changed. We have been changed. Emblazoned in our collective consciousness are indelible, fresh images of unspeakable carnage. We will carry these images with us – shaped in psychic scar tissue – for the rest of our lives.

For some of us, September 11 will mark the date we lost an acquaintance, a colleague, or a loved one. For many of us, especially our children, it will mark the date we lost a certain degree of innocence. For all of us, it will mark a dark milestone on the landscape of our memories.

My 11-year-old son came home from school that Tuesday with an unsettling bloodlust in his eyes. He had seen the events unfold and was having a purely visceral reaction. Decorum prevents me from recounting all his suggestions for acts of retaliation, but suffice it to say that the U.S. position as a nuclear superpower figured prominently among them.

I tried to settle him down and to explain that it would be difficult to avoid hurting innocent people if we just reacted without thinking, but his frustration and anger were beyond attenuation. He fumed and boiled and spouted epithets that surprised even my salty sensibilities.

However, just below the surface of his preadolescent bluster and fury, I could sense a stifled cascade of terrified tears. I felt powerless to console him, and that angered me. He'll never look at the sky the same way again, and neither will I.

I tried to remember what it was like when I was his age. Oh sure, I had reasons to be afraid when I was a kid, too, but they were a little less omnipresent than the chance sighting of a commercial airliner. The sparks that fueled my fears were a little more "abstract" than that, though the fears themselves seem every bit as real in my recollection.

I grew up a few miles west of a U.S. Air Force base, and although I'm a little too young to remember the "duck and cover" drills my older brother had to practice, I do remember growing up with the perpetual fear of nuclear conflagration. My nightmares emanated from a bright white light that flashed suddenly on the eastern horizon. The few minutes of life left to me, before that light would inevitably erase all vestiges of my little world, seemed to pass by so very slowly. It was a relief to wake up in the darkness.

My father was a professor at a local college, and during the summer months he taught a course on fallout-shelter analysis for the government. He tried to ease my fears, too, by telling me that our little Air Force base would be among the last targets in any real "shooting war." Somehow, as I remember, his words didn't help all that much. It was the unquestioning and unconditional love in his eyes that was the greatest comfort. Logic took a back seat to emotion, I guess, even then. Some things don't change.

With the dissolution of the Soviet Union, fears of nuclear holocaust don't haunt me in quite the same way they used to. Now, though, a new set of fears has been unwrapped and left on my doorstep. And, for the first time I have to try to protect my own children from the burden of living in fear.

Perhaps it is our reaction to fear that will measure us. There's an old saying that goes something like this: "Without fear, there is no courage." Well, I can tell you we have a renewed patriotic courage in the U.S. – a courage and unity I haven't felt since the wave of grief that swept over America and the world in November of 1963.

Apparently, we've found evidence that points to Osama bin Laden as one of the key architects of the attacks on New York and Washington, DC. If we can bring this man to justice, perhaps we will be able to regain some tiny iota of our lost security. However, the 19 direct perpetrators are beyond any human sanctions now, and that is a consummate frustration.

I sincerely hope our allied response to these attacks will be effective in preventing them in the future. If the fanatics who planned these attacks intended to strike fear into our hearts, to some extent they have succeeded. My son's fear is evidence of that. However, our fears are aroused at a terrible price. Prepare to pay.

AUTHOR BIO
Blair Wyman is a software engineer working for IBM in Rochester, Minnesota, home of the IBM iSeries.

blair@blairwyman.com

Java Dudes