NEW Forte™ for Java™ 3.0 tools from Sun. Review: page 56

JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

September 2001 Volume: 6 Issue: 9

JAVADEVELOPERSJOURNAL.COM

FREE NEW Download Forte™ for Java™ 3.0 tools
Sun microsystems
www.sun.com/forte/ffj

RETAILERS PLEASE DISPLAY UNTIL NOVEMBER 30, 2001
$5.99US $6.99CAN

SYS-CON MEDIA

JAVA DEVELOPER'S JOURNAL

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

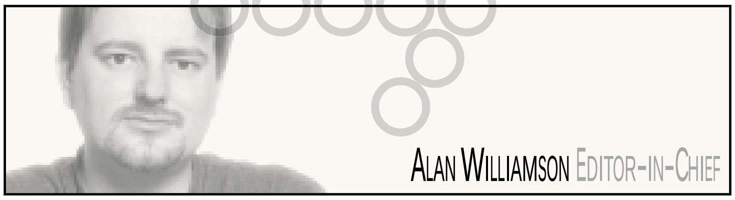
PRODUCTION
VICE PRESIDENT, PRODUCTION AND DESIGN: JIM MORGAN
EXECUTIVE EDITOR: M'LOU PINKHAM
MANAGING EDITOR: CHERYL VAN SISE
EDITOR: NANCY VALENTINE
ASSOCIATE EDITORS: JAMIE MATUSOW
GAIL SCHULTZ
BRENDA GREENE
ASSISTANT EDITOR: LIN GOETZ
TECHNICAL EDITOR: BAHADIR KARUV, PH.D.

WRITERS IN THIS ISSUE
CHARLES AREHART, JOSÉ MARIA BARRERA, JASON BELL, ZEV BLUT, JASON BRIGGS, GLENN COATES, MARK DYKSTRA, THOMAS HAMMELL, GLEN MARTIN, JAMES MCGOVERN, CHRIS MELISSINOS, JIM MILBERY, VINCENT PERRIER, VIJAY RAMACHANDRAN, STEVE ROSS-TALBOT, AJIT SAGAR, TOM SLOPER, JASON WESTRA, ALAN WILLIAMSON, BLAIR WYMAN

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.

WORLDWIDE DISTRIBUTION BY:
CURTIS CIRCULATION COMPANY
NEW MILFORD, NJ



Together We Stand, Divided We Fall

This is the time of year when most people take their vacation…when productivity falls a little below the yearly average…when nothing exciting happens. Mainstream journalists call it the "silly season"…the time when they usually have to dredge up all those human interest stories about terribly dull people doing extraordinary things.

Our news world also suffers a little from this vacation syndrome, with no earth-shattering stories to keep ahead of. This year hasn't been so bad, however – thanks largely to Microsoft and the whole debacle surrounding Java and Windows XP. As my esteemed colleague Jason Briggs writes in his editorial, we've had a lot of fun reading what can only be described as propaganda surrounding the so-called demise of Java and how the world is now going to move toward C# as an alternative.

What a lot of nonsense! Don't panic. Java is safe and will continue to be a major force in software development. Incidentally, for a wee bit of fun, check out our back-page cartoon. Our editors do have a wicked sense of humor.

As you know, I'm based in Scotland and travel to New York regularly. Each time – conservative spender that I am – I look for the cheapest flight. Nine times out of 10 this is Icelandic Air, and Reykjavik airport is such a wonderful place to stop over in, breaking up the journey nicely (stick with me – my story gets better).

What I like most about using Icelandic Air is that their whole Web site is based on Java, and it gives me a warm feeling to be actively using the technology that we spend the majority of our time talking and writing about. Admittedly they're using JSP, which makes me cringe, but hey, you can't have everything.

Of late, however, the team up in Reykjavik has been having problems keeping the site up. In fact, as I write this editorial, the site is unable to accept bookings and I've had to resort to flying with Mr. Branson at Virgin. I've been in constant dialog with the team at Icelandic Air, probing to see what's going wrong, because as much as I hate JSP, I hate seeing such a high-profile site that's using Java not be available. It does nothing for Java's reputation as a whole, and it's important we don't give Mr. Gates any more excuses than he already has for criticizing Java's performance.

Icelandic Air is using a number of high-profile products to host their JSP offering – their names will remain anonymous. The problem was sourced to a start-up script that failed to load one of the servers in their server farm, rendering the whole farm useless. Fortunately, this has been fixed (so the problems I'm seeing now must be new ones!).

• • •

I've been fortunate to meet and know many powerful women in the world of computing. On the whole, getting where they are hasn't been easy. They've generally had to work twice as hard as their male counterparts to prove themselves. We all know it shouldn't be this way, but that's the harsh reality of the corporate (man's?) world. In much the same way, I can't help but draw a parallel with Java proving itself in the corporate world against all the alternatives.

When problems do occur, I've seen upper management quick to fault the language and lay the blame at Java's doorstep, muttering that "we should have gone with such-and-such." Because of its high-profile marketing, Java is an easy target. And thanks to the days of the "gray-box-in-a-browser," when a site is running slow, the reason given is often Java.

We all know this is a lot of nonsense and usually just masks the real reason something is going wrong, which in the majority of

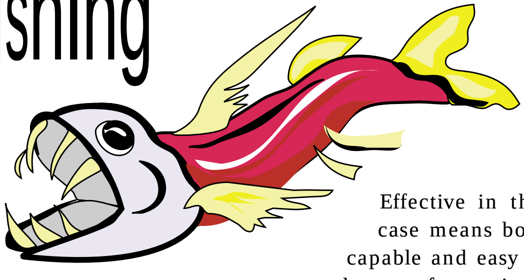
–continued on page 78

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 GLEN MARTIN

Gone Fishing



I saw a television ad the other day that portrayed someone using a cellular phone as a fancy cash card to make a vending machine purchase. As a person who hates to carry loose change – once you start, you suddenly realize you have a pocketful – this spoke to the kind of useful integration into people's lives that a new technology needs to be successful.

If you think a cash card is easier to carry than a cellular phone...you're right. But the integration of functions into one device is more convenient still.

A cell phone is one device that could act as a cash card, map, traffic-status monitor, voice recorder, and yes, a competitive deep-sea fishing game.

That last one isn't a joke, by the way. I hear it's the most popular downloadable content in Japan. Apparently the phone vibrates with the tension on the line.

Everyone agrees the wireless industry is poised for incredible growth over the next couple of years.

More interesting is that by 2006 wireless may well supplant wireline as the dominant means to access services. This means there will be an increasing number of users who <i>only</i> have wireless access. And that access will be fast – even 2.5G has a bandwidth comparable to the DSL lines many of us use today.

An examination of the wireless applications being developed shows the network and back end as critical elements of the overall solution. E-mail, consumer e-commerce, and location-based services are a few examples.

Even a deep-sea fishing game needs to synchronize scores, and perhaps simulate the fishing lines getting tangled when 20 suits on the same Tokyo street corner hook a marlin at the same time.

This begs the question: "What infrastructural capabilities are required for the server-side of wireless applications?"

Scalability is clearly most important. Many people will be using handhelds to access a wide variety of services.

But there are less obvious requirements.

Many of these applications aren't <i>pure wireless</i> – they need to access back-end applications and data. So effective integration with back ends using a standards-base architecture is critical to the success of the wireless application. Effective in this case means both capable and easy to program, because faster time-to-market for both new applications and upgrades will be important to profits.

These back-end applications don't all run on one platform. Consumer e-commerce applications will need to be integrated with billing and shipping systems, some of which have been around for many years, and deployed on a wide variety of hardware.

Another reality is that consumers will access the same services with a multitude of devices, from palmtops to phones and even the desktop. A large online book retailer lets me browse and buy books from my desktop and palmtop – hopefully using the same application code.

Finally, these applications will see rapid upgrades. We're entering a world of mix-and-match services in which new applications will be aggregated out of existing services as quickly as people can imagine it. Sometimes these applications will be transactional, as in e-commerce. This leads to the requirement for a flexible component reuse platform that allows <i>just right</i> integration.

We're looking for a highly scalable platform with great support for back-end integration, device- and platform-independence, and flexible component reuse that supports transactions.

This is where the J2EE fits in. The Java technology community has been solving these problems for years now, and J2EE version 1.3, released in September 2001, has the solutions in a 3G platform.

In particular, the J2EE Connector Architecture provides frequently demanded standardization to the integration of back-end systems into new applications. This is what will eventually make these new wireless applications useful, and foster the adoption of the next wave of wireless applications.

The J2EE Client Provisioning Specification, currently in process, will go one step further to manage and deploy the different client-side parts of applications that serve multiple devices.

Even better, J2EE eliminates vendor lock through strong third-party vendor support.

Check it out.

Gotta run, I've got a fish on my phone.

glen.martin@sun.com

AUTHOR BIO

Glen Martin is the J2EE Specifications marketing manager at Sun Microsystems, and is responsible for specifying and advocating the future of enterprise Java. He has 14 years of industry experience building everything from packet switches to development tools.

magicdraw™

The rational alternative

Version 4.5
Introductory Offer
$2995
for Teamwork Server

All 9 UML diagrams

Additional Features:

- Performs Java, C++ or CORBA IDL code round-trip engineering (code generation and reverse engineering); recognizes JavaDoc comments. This feature allows you to write code, reverse engineer, make changes to the model and re-generate the code without losing any implementation specific information.
- Supports UML 1.3 notation.
- Saves diagrams as bitmap PNG/JPG and scalable WMF/SVG/EPS/DXF formats.
- Provides XML interoperability – native model files are stored in XMI format.
- Integrated with Forte for Java (FFJ) IDE versions 1.0 and 2.0.

Standard Edition: $249
Professional Edition: $499

Floating License . . $100 additional
Upgrade for a $69 annual fee!

No Magic... Just Technology That Works

Download fully functional demo at:
www.magicdraw.com

contact us at
contacts@magicdraw.com

100% Pure Java Application

Comments on Jon Stevens Article

I enjoyed Jon Stevens article ["JSP, You Make the Decision," Vol. 6, issue 7].

I'm frustrated by all the limitations outlined in the article. I am interested in the Velocity template solution; where can I find more information on the Velocity/Turbine project? I'm tired of "fighting" with JSP and all its marketing hype; it's not really a good MVC solution for complex Web applications!

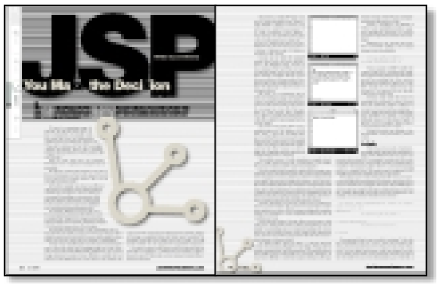
Mark A. Sellers
Mark.A.Sellers@wcom.com

P.S. The above is my opinion so don't blame WorldCom!

I hated the article from Jon Stevens on JSP alternatives [Vol. 6, issue 7].

He did not discuss Velocity or its intended purpose, but jumped into picking apart JSP against Velocity. He focused solely on the benefits of Velocity, and then hoped we enjoyed his tour of "alternatives." I felt like I was reading one giant anti-JSP gripe-fest.

His examples of JSP code are contrived, purposely written poorly to make JSP look weak.



Having worked in ASP, Perl, ColdFusion, and JSP, I can say that JSP cuts my coding time by more than half, through the use of tag libraries, etc.

My biggest beefs are:

1. Jon talks about the transform->compile process in JSP, saying that it has to recompile the JSP each time a request is made, which is blatantly untrue on many platforms. Maybe when Jon was still coding JSP in 1998 that was true.
2. Jon also says that JSP breaks the vaunted MVC design and you have to embed HTML in the code, but then goes on to show us Listing 7 in Velocity, which clearly has HTML embedded in the middle of his code. What's with the double standard?
3. Jon speaks about how HTML programmers won't understand notions of scope, or grasp complex programming issues. True, most HTML designers don't know about coding more complex applications. So don't let them do it! Many shops I've worked with complete the design and HTML framework, the basic templates of the site, and then the back-end programmers complete the inclusion of the page logic.

Next time if you're going to talk about alternatives to JSP and discuss the problems in JSP, include Cocoon, Velocity, and other Java solutions, and get someone with a more open mind to evaluate them.

Maurice Reeves
Maurice@pipelineInteractive.com

I just read Jon Stevens article about JSP, Struts, Velocity, etc. I was really pleased that I'm not the only one that doesn't like JSPs that much. Great article!

Couldn't have said it better... :-)

Andreas Prohaska
ap@apeiron.de

The article by Jon Stevens on JSP was the most unbelieveable and unadulterated rubbish that I have ever read! (perhaps excluding Microsoft press releases). Not only were his arguments vacuous and just plain wrong, he showed he had absolutely *no* understanding of the issues surrounding the two concepts (push versus pull) whatsoever. I am very, very disappointed in this article.

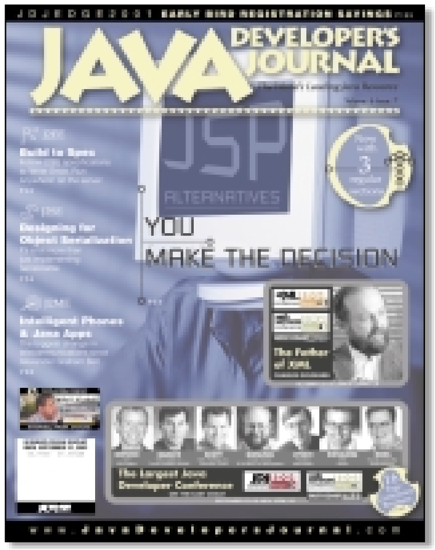
Richard Vowles
rvowles@borland.com

I am interested in what exactly you think is unbelievable about what I wrote. I attempted to stay very technical and detailed and the points against JSP are actually valid points. What parts of my statements did you find vacuous?

I'm also interested in what part of the Pull versus Push model you think I don't understand. I wrote a document that has been available on the Jakarta Turbine Web site (http://jakarta.apache.org/turbine/pullmodel.html) for over seven months now that clearly shows my understanding of the Pull model. If there's something inaccurate about what I said, I would appreciate you clueing me into what is wrong so that I can correct it.

Last, I would appreciate it if you would take a few minutes and evaluate Velocity. The reason is, I understand that it's difficult to have someone stand up and point out faults in a technology that corporate marketing is telling everyone they should be using. However, I believe it's always good to keep an open mind, an open view, and a lookout for other technologies that may actually do a better job than the ones pushed on us.

Jon Stevens • jon@latchkcy.com



EJB-Based Services

"Build to Spec" by Liz Blair is a wonderful article [*JDJ*, Vol. 6, issue 7]. It provided excellent insight into differentiating between and architecting with the many components of J2EE.

Vince Huston
vince.huston@valtech.com

Need an Investor?

I enjoyed Jason Briggs's article, "A Beginner's Guide to Writing Applications for the MID Profile" [Vol. 6, issue 7]. I would like to know if you're planning on having a second, third, or *n*th part to the above article with more practical and realistic examples? If the answer is no, could you recommend a book on this subject?

And good luck on your new Java-enabled device to take over the new generation. Let me know if you need an investor.

Mehrdad Shabestari
Mehrdad.Shabestari@htsco.com

JavaOne Show Review

Ajit Sagar's article was very well balanced and nicely done. I attended both 2000 and 2001 and had the exact same impressions. His review was in sharp contrast to that of a competing Java periodical, which was so rosy as to make me wonder whether they had been at the same conference at all.

Mike Silverstein
msilverstein@silvermark.com

Thanks, Mike. The effects of the economy were fairly obvious. Besides the attendees, I also chatted with the folks at my hotel and a couple of cab drivers. They are the best sources for sensing the pulse of such events. I'm also glad to hear that JDJ *was able to cover the event to the satisfaction of readers like you. Letters such as yours make it all worthwhile.*

Ajit Sagar • ajit@sys-con.com

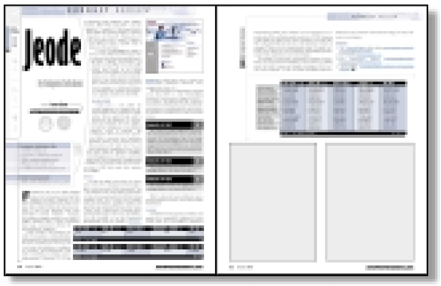
Insignia on Jeode

The review of the Jeode platform [***JDJ***, Vol. 6, issue 8] is welcome exposure for Insignia Solutions, but the context in which our product was evaluated doesn't give your readers a representative picture of the product's true performance attributes. While pitting the Jeode Embedded Virtual Machine (EVM) against four JVMs employing Just-in-Time (JIT) compilers in a memory-rich NT desktop environment makes for an interesting race, it doesn't reflect the real choices developers have when deploying a JVM in a memory-constrained device.

Jeode technology works well on NT but is not intended for the desktop environment. However, in constrained environments, the dynamic adaptive compilation (DAC) approach that the EVM employs has proven to be much more effective than interpreter-only or JIT compilation approaches.

In benchmark tests where memory resources are freely available – like those conducted by ***JDJ*** – JIT-based JVMs will shine because they can fully compile a small application at start-up so that the whole application runs as native code. But this is not representative of real applications, which are much bigger and dynamic. So while JIT-based solutions can deliver fast performance for some small applications, particularly benchmarks, they do not scale to larger, real-world applications on resource-constrained devices.

DAC technology delivers the best possible performance in memory-constrained devices, because it compiles the most frequently executed code without incurring the overhead penalties associated with compiling infrequently executed code. It's competitive with desktop JVMs employing JIT compilers in a NT environment, but its performance attributes are best observed in the native resource–constrained environment for which it was designed.



Gary M. Katz, APR
Senior Manager, Corporate Communications
Insignia Solutions
gary.katz@insignia.com

While I might agree with Insignia's arguments regarding comparisons between DAC and JIT technologies, I stand by the review. Jeode was also tested on a Compaq iPAQ (as part of the iPAQ review) – exactly the type of resource-constrained environment mentioned – so it was an interesting comparison to look at its performance against JVMs in a completely open environment (memory-wise, that is). I think the fact that Jeode held its own against the "big-boys," in an environment it is not suited for can only be seen as a selling point.

Jason Briggs • jasonbriggs@sys-con.com

Helen Thomas's Article

I read Helen Thomas's article "Accelerating Java Web Application Environments" [Vol. 6. issue 7] and wanted to tell you how I enjoyed it. Although her article focused on wired dynamic Web sites as apposed to the wireless Internet, I'm very interested to see how these spaces evolve together as Internet standards continue to develop and true 3G technologies catch up with developer's applications.

Kirsten Brundahl
kbrundahl@bockpr.com

New Look

The new look of ***JDJ*** doesn't work.

1. Many articles no longer include listings, which makes reading the articles impossible unless [you are] seated next to a monitor.
2. The new tendency to print text in "artistic" colors like lavender, olive, and pale gray results in text that's nearly illegible.

My favorite monthly magazine has "evolved" into a chore to read. Please stick to high-contrast color schemes.

Alan Wolfson
alan@dotobject.com

Thank you for that and I will pass on your comments to the production department. But you mention nothing of the quality of content, which has gone up. How has that fared with you?

Alan Williamson • alan@sys-con.com

But I Love the Content

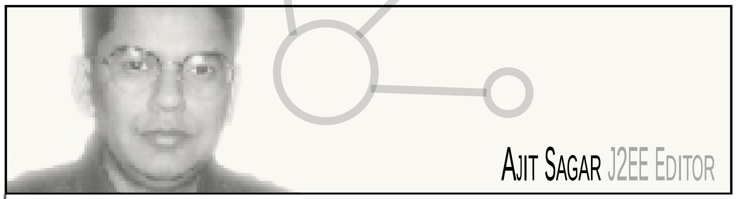
I was trying to keep my note short and sweet. I love the content, and recognize the changes in having essentially three separate magazines between the covers. There's no question about it being at the top of my "to read" pile, and I get one of everything, including *JOOP* and the lesser-known mags.

You used to do a fantastic job of printing edited listings that illustrated the article. It used to be a joy to read the only magazine that printed enough code to support the context of the articles.

I do like the color schemes for various articles. Just use high-contrast combinations for the text/code.

Alan Wolfson
alan@dotobject.com

The views and opinions are those of the readers and do not necessarily represent the views and opinions of their employers.



AJIT SAGAR J2EE EDITOR

Thinking Outside the Box ...

They say no man is an island. For J2EE I would say no platform is the universe. Sometimes folks misunderstand the promise of J2EE. It won't replace every other development paradigm. J2EE application servers won't make all other deployment and runtime environments obsolete. And Java won't be the only language people program in. Java's strength is facilitating development, but it isn't the first and last thing of the programming world. You can use J2EE to build enterprise applications, but you can't use it to build the enterprise itself.

The distinction is important. When you write enterprise applications, you model a portion of the enterprise. You basically take your organization/division/group's core technology offering, identify the section of an enterprise's business process that you choose to model, and then go about building the application.

If J2EE is your platform of choice, and I hope it is, you build the business components and the application logic using a J2EE application server and J2EE APIs. However, this application provides a piece of the bigger puzzle. The rest of the application environments you connect to may use J2EE too, or a portion of it, or be totally independent of J2EE.

So how do you connect to non-J2EE environments? Well, depending on the boundaries of the "J2EE box" you've identified, you define the integration points. You then use a combination of the facilities offered by the J2EE platform and other cross-platform technologies to talk to the outside world.

At the front end these technologies could include servlets and XML. These APIs allow you to present to the client in a platform-neutral way. At the back end J2EE offers connectivity to third-party systems through various technologies. J2EE provides connectivity to enterprise information systems (EIS) through the recently released Java Connector Architecture (JCA).

With JCA your application can step outside the J2EE box to conduct business with third-party legacy systems. The vision of JCA is that standard connectors will be developed that allow Java environments at one end to communicate with legacy environments at the other end. However, JCA implementations are still immature and the resource adapters still need to grow up. JCA currently competes with more offerings from mature vendors – such as WebMethods and Tibco – that directly connect to ERP systems and already have sets of predefined adapters. The only catch is that when you use these vendors' offerings, you also end up using their proprietary execution environments.

The term *proprietary* is relative. The moment you choose a particular application-server vendor and a particular resource-adapter provider, you end up using particular implementations. But, as an enterprise developer, the good news is you continue to develop using standard Java APIs that can apply across different vendors.

Since the J2EE application-server vendors are extending their containers to support connectivity via JCA, we can hope that a couple of years later the connectivity layer to EIS becomes a standard commodity. The actual realization of this promise, of course, will take a bit of work. Plus it will be a hard sell for a company to bundle both vendors' offerings to a single customer.

An interesting development in the marketplace is the partnerships emerging between the J2EE application-server vendors and the EIS connectivity providers. One of the primary partnerships recently announced was the one between WebMethods and BEA that allows Java

–continued on page 78

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.

J 2 E E I N D E X

WHAT TYPES OF APPLICATIONS CAN YOU BUILD WITH J2EE TECHNOLOGIES?

The J2EE platform can be used to build enterprise applications (from Java 2 Enterprise Edition). Enterprise applications are inherently distributed.

In terms of the Java platform, this means that the components of the applications run on different JVMs that may be distributed across various machines on a network. Since the common mechanism for communicating between processes running on different JVMs in Java is Remote Method Invocation (RMI), this means RMI is the preferred communication protocol for distributed communication in a J2EE application. Since J2EE supports the Web paradigm, however, communication between the presentation layer of the application and the business-tier components commonly uses HTTP as the communication protocol. The J2EE platform is designed to provide server-side as well as client-side environments for developing multitier enterprise applications.

DO ALL APPLICATIONS BUILT ON J2EE REQUIRE A WEB INTERFACE?

This is a common misconception. Since J2EE defines the servlet and JSP APIs for Web-based access, some people think that J2EE is meant only for applications with a Web-based front end. It's not true that J2EE applications can be accessed only via a browser. J2EE applications support a variety of clients, including wireless and small devices, rich Java clients, and non-Java clients.

The confusion arises from the fact that thin Web clients are an increasingly popular way of building application front ends. That's why JSPs and servlets have gained popularity in Java. As a result, Java-based J2EE applications are often exclusively associated with Web applications.

DO ALL J2EE APPLICATIONS REQUIRE EJBS?

The short answer is no; not all J2EE applications require EJB development. That said, EJBs are the crux of J2EE. The J2EE application platform is built around the concept of EJB-based business objects. In any distributed programming platform, the middle-tier business components are built on a common software object component model. COM is the component model for Windows-based applications. EJB is the object model for J2EE applications.

Essentially J2EE applications require application containers. In last month's FAQs (***JDJ,*** Vol. 6, issue 8), the different containers supported by the J2EE platform – Web containers and EJB containers – were discussed. A Web-based distributed application that leverages J2EE technologies to the fullest will make use of both containers: the presentation components run in the Web Container and the business components executed on the EJB Container. All the business components are built as EJBs. The client accesses the application via servlets or JSPs. The information can be exchanged in the form of XML/HTML.

However, this is not the only application scenario. Remember that J2EE defines other APIs besides the servlet, JSP, and EJB APIs. The ones under the spotlight for non-EJB applications are RMI, JDBC, and JNDI. Application components can use RMI to communicate between distributed Java objects (which don't have to be EJBs). JDBC can be used directly from either servlets/JSPs or a Java client to access back-end data sources, and JNDI can be used for binding to remote objects via a directory service.

A typical Web-centric application can be built using just servlets/JSP pages and JDBC for database connectivity – with EJBs completely out of the picture. This still falls under the category of a three-tier distributed application (with the business logic housed in the servlets layer. Remember that JSPs essentially equate to servlets at runtime). The access to the data source can also be decoupled from the servlet layer by moving the data access logic to another Java object. This Java object can reside on a separate machine from the servlet and the communication can be facilitated via RMI.

In short, J2EE can be leveraged to build non-EJB applications in many ways. The true benefits of J2EE, however, are reaped when you design your business-logic layer on EJBs. J2EE application servers provide the environment for designing and executing EJB components, and only when you leverage the component model supported by J2EE will you benefit from the development environment that it offers. Without EJB, you'll end up doing a lot of the work made obsolete by the emergence of J2EE application servers as the "OS of the enterprise."

Correction in August FAQ: JNDI stands for "Java Naming and Directory Interface," not "Java Native Directory Interface." Thanks to Krishna Kota for catching the error.

J2EE ROADMAP

The Java 2 Platform, Enterprise Edition defines the APIs for building enterprise-level applications.

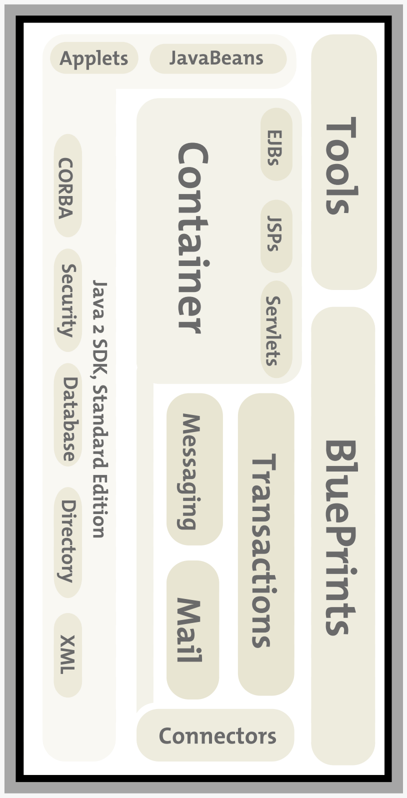
J2SE	v. 1.2
Enterprise JavaBeans API	v. 1.1
Java Servlets	v. 2.2
JavaServer Pages Technology	v. 1.1
JDBC Standard Extension	v. 2.0
Java Naming and Directory Interface API	v. 1.2
RMI/IIOP	v. 1.0
Java Transaction API	v. 1.0
JavaMail API	v. 1.1
Java Messaging Service	v. 1.0

Useful URLs:
Java 2 Platform Enterprise Edition
http://www.java.sun.com/j2ee/

J2EE Blueprints
http://www.java.sun.com/j2ee/blueprints

J2EE Technology Center
http://developer.java.sun.com/developer/products/j2ee/

J2EE Tutorial
http://java.sun.com/j2ee/tutorial/



JCP Defines the Roadmap for J2EE

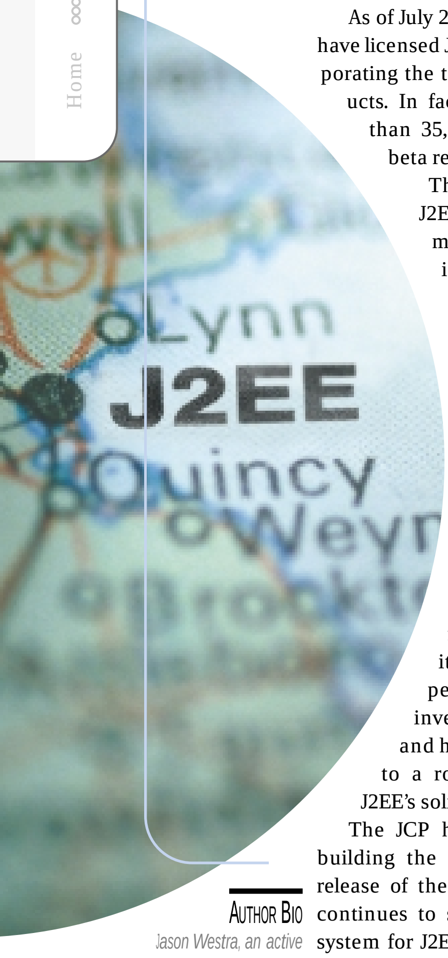
WRITTEN BY
JASON WESTRA

I never bothered with roadmaps until I was of driving age and began to take trips on my own. Rock climbing drew me to my first trips and involved driving to remote areas of the U.S. It didn't take long to realize that a single wrong turn onto a road in the middle of nowhere meant hours of wasted time. Too many wrong turns soon earned you the nickname Clark Griswold. A wrong turn was especially a problem back in the days when Montana had no speed limit. You could really get nowhere fast back then.

"Nowhere fast" is exactly how I felt when developing distributed applications with Java before the J2EE platform. Originally labeled the Enterprise Java APIs in 1998, the platform was introduced by the Java Community Process (JCP) led by Sun Microsystems and a core team of JSR expert group members.

Since then, the platform:

- Has been renamed Java 2, Enterprise Edition
- Has incorporated many new APIs
- Has gained great popularity among the masses of enterprise developers



As of July 2001, more than 27 vendors have licensed J2EE and are actively incorporating the technology into their products. In fact, there have been more than 35,000 downloads of the 1.3 beta release alone.

The current release of the J2EE specification (1.2) has made great strides in moving server vendors from providing half-solutions to providing full platforms for enterprise development. The specification essentially glues together the numerous Enterprise Java APIs, such as EJB, JSP/Servlet, JDBC, JTA/JTS, JNDI, and JavaMail. This movement toward a standard platform has fostered portability across vendors and peace of mind for businesses investing in Java technology, and has guided many a wayfarer to a robust solution built from J2EE's solid architectural blueprint.

The JCP has been instrumental in building the roadmap for the current release of the J2EE specification and it continues to shape the future highway system for J2EE 1.3 and beyond. In case you haven't heard, JSR-58 released the final public draft of version 1.3 on April 9 of this year. It includes some great new features, such as required support for JMS (Java Message Service) and EJB 2.0 (JSR-19), including the new message-driven bean, which takes advantage of the JMS APIs. The J2EE Connector Architecture 1.0, also driven by the JCP (JSR-16), is mandatory, and will allow server vendors to seamlessly connect to enterprise information systems with resource adaptors that manage resource pooling, transactions, and connectivity issues.

The J2EE 1.3 specification included requirements for IIOP and XML, emphasizing portability across the enterprise and between enterprises. The final release of this specification is Q3 2001, and while I look forward to its release, I'm already looking at the future roadmap of the platform.

A gander at the JSR list on JavaSoft's Web site (http://java.sun.com/aboutJava/communityprocess/search.html) will show you where the action is.

While I already noted some of the JSRs that have influenced the J2EE specification to date, many more exciting JSRs are looking to shape the future of the platform. Work is already underway to firm up loose ends with the J2EE Connector Architecture. JSR-112 will have a Community Draft available in Q4 2001. Unfortunately, this means we'll probably have to wait until J2EE 1.4 or so to see its inclusion.

An interesting effort is taking place to define management of J2EE applications. JSR-77 is defining a management model that will:

- Allow management of heterogeneous vendors from a single tool
- Provide integration with existing management systems
- Allow for a single tool to manage heterogeneous vendor implementations

One of the APIs many feel is missing from the J2EE Platform is JMX (Java Management Extension or JSR-3). However, the J2EE platform architects have expressed a more open approach toward management in the J2EE specification and don't directly support JMX over other management technologies. Thus JSR-77 looks to provide standardized management for J2EE applications by including support for existing technologies such as SNMP, JMX, and WBEM. JSR-77's time line is for a Proposed Final Draft in October 2001, with inclusion in the J2EE 1.4 release of the specification.

Next, the mission of JSR-88 (my personal favorite) is to determine requirements for the portable installation and configuration ("deployment"), and removal ("undeployment") of J2EE modules across J2EE servers. J2EE has been adopted at an alarming rate, and enterprises with multiple J2EE vendors have discovered problems deploying applications across their heterogeneous environments. The J2EE specification doesn't address portable deployment of applications across heterogeneous servers; however, with the inclusion of the J2EE Deployment API into the J2EE specification, portable deployments will become reality. JSR-88's milestones are a Public Draft around Q2 2001, and inclusion into J2EE 1.4 when it's released.

Last, JSR-117 (J2EE APIs for Continuous Availability) looks to be an important contributor to the J2EE specification of the future. It will address shortcomings in the current specification around ensuring high availability of J2EE applications, including defining a portable API for failover management, online upgrades of J2EE components, logging conventions for system administrators of J2EE applications to better monitor and read system reports, and error management for system-level exceptions so as to offer automatic recovery of state and transactions. Once this JSR is finalized, it will prove to be a powerful inclusion in the J2EE specification.

As both *National Lampoon's* Griswold and I know all too well: "Getting there is half the fun!" So if you ever feel you need a roadmap to point you in the right direction, take a look at maps the JCP is providing. If you have the urge to participate in the JCP, don't be shy. Anyone can be a member and contribute to the J2EE specification or other technology roadmaps of the future. Why not you?

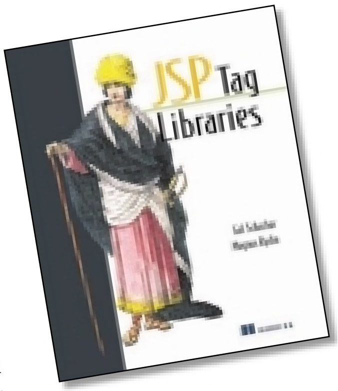
jason@sys-con.com

AUTHOR BIO
Jason Westra, an active member of the Java Community Process, is currently on the expert group for JSR-88, J2EE Deployment API.

JSP Tag Libraries

JSP Tag Libraries
by Gal Schachor, Adam Chace, and Magnus Rydin
Published by
Manning Publications Co.
Price: $44.95

REVIEWED BY JAMES MCGOVERN
jmcgovern@enherent.com



Tag libraries were introduced into the Java specification to solve many of the limitations of using scriptlets (bits of Java code) as part of a JSP page. The main limitation is that advanced page design may require the designer to understand Java to perform tasks such as constructing a loop, if/else blocks, and sending an e-mail, and so on. Art designers and HTML developers are creating many pages in today's Web development environment, making this approach less than optimal. Tags provide a mechanism that will allow a non-Java developer to utilize Java functionality without having to learn the language. Tags can also assist Java developers in separating presentation from control. Tag libraries were introduced as part of the JSP 1.1 specification.

JSP Tag Libraries is published by a small book company (Manning Publications) known for writing books of very good quality. As this publishing company is smaller than the Wroxes and Addison-Wesleys of the world, you may not find it on the local bookstore's shelf, and may have to order it through Amazon or Fat Brain. Reasonably priced at $44.95, it has about 600 pages, divided into 15 chapters. This book doesn't waste your money by selling you freely downloadable Javadocs.

As I started to read, I noticed that the book was very well thought out. It contained only minor typographical errors and had a good flow. This book is not sloppy like others I've read in the past, and I appreciate that the authors actually understand how to use the English language. It doesn't read like a manual but more like a conversation, developer to developer.

Part 1 begins with a light introduction to Web development using Java. It covers how JSPs are developed and goes into sufficient detail on their strengths as well as their limitations. Later chapters cover a tag's life cycle and how it may be used. Typically, an experienced developer may decide to skip over this section, but I recommend reading it, as it also has some hidden gems on JSPs.

Part 2 covers basic techniques on how to develop a tag. The first useful tag that is developed is for sending e-mail. Other tags that are introduced cover interaction with a JavaBean, using tags for validation, assertions, control of flow, as well as accessing the application's back end. Chapter 8, one of the best chapters, covers writing a tag that discovers a JavaBean's methods and events at runtime (reflection), a useful technique for all tag developers.

Part 3 dives into advanced techniques for creating conditional tags to insulate HTML developers from the tricky syntax of creating if/else and loops using scriptlets. Chapter 11 introduces usage of tags with Model 2 architectures. Chapter 12 focuses on using tags in a J2EE environment, and covers usage of EJBs and integration with business logic. There is an awesome discussion on using tags along with JNDI. Within this chapter, two really powerful tags are introduced, one for looking up a bean's home and remote interfaces, and another for connecting to a JDBC data source.

Part 4 covers two different case studies: building a JDBC-driven Web store and an EJB-driven WAP store. Part 5 closes the book nicely with architecture and design advice along with a healthy dose of tips and tricks that will help make your applications reusable, maintainable, and scalable.

I could find only a single flaw in the book: I was hoping there'd be mention of other tag libraries that are a part of many application servers such as WebLogic or JRun, as their libraries are pretty good. I'd especially love to see a chapter or two on the Jakarta tag libraries in a future edition of this book.

JSP Tag Libraries should be part of every developer's library and is worthy of being on the same shelf as *UML Distilled*, *Design Patterns* by the Gang of Four, and *Inside Servlets* by Dustin R. Callaway. This is the first book on this topic that not only covers the nuts and bolts of using tags, but also shows how it can fit into your overall architecture. I look forward to other books by these authors and would recommend them without delay. I rate *JSP Tag Libraries* only one-quarter star shy of a possible five stars.

Next I'll review *Effective Java* by Joshua Bloch, published by Addison-Wesley. James Gosling, father of the Java language, mentioned this book during his keynote speech at JavaOne. Let's see if it lives up to its name.

Making the Move to J2EE

Java and J2EE resources

WRITTEN BY
CHARLES AREHART

Welcome to the first installment of Journeyman J2EE. I'm honored to present this new bi-monthly column of ruminations and reactions as I, like so many of you, make my foray through the vast world of J2EE application development and deployment. But this isn't intended just for newbie J2EE developers. On the contrary, I hope to also share tips and techniques of value to experienced JSP/servlet developers.

The definition of a journeyman is "a competent and reliable performer or exponent." To me, it connotes a day-to-day working craftsman.

Does this describe you? Are you like a journeyman infielder in professional baseball (apologies to international readers)? It's not that a journeyman isn't valuable to the team. Journeymen contribute in a professional, competent, workman-like way, and strive to improve their abilities and refine their craft.

That's the goal of the **Journeyman** column, which began nearly two years ago as a monthly column in our sister magazine, ***ColdFusion Developer's Journal***: to share tips and techniques that aren't quite advanced, but aren't quite beginner either. With four years as a Web application developer (mostly in ColdFusion), trainer, and writer, and 20 years of IT experience, I hope I'm in a position to discover and share useful observations.

Filling a Gap for Both Experienced and Newcomer J2EE Developers

Some of you may scoff at my mentioning experience in ColdFusion (which is really a shame, since it's often unfairly maligned). The bottom line is that it's a Web application development platform, just as ASP, PHP, and J2EE are. Admittedly, each has its own distinct flavor and unique capabilities.

But when it comes down to it, there really are quite a lot of similarities among all Web application development platforms: HTTP processing, opportunities for client enhancement, clever form-processing tricks, effective database integration, session processing, and lots more.

As I move into the J2EE arena I notice that quite a few techniques and approaches used by developers on those other platforms haven't made it into the toolbag of many J2EE developers. Perhaps it's because the focus for them has been more on simply getting into Web apps as they move from Java client to server development. But we who develop on those other platforms have been doing Web apps for years, playing all manner of tricks with browsers, sharing data between Web sites, and more. Maybe these experiences can be of value to J2EE developers.

So on the one hand, the column will speak to experienced J2EE developers by offering Web application development ideas that may be new to them.

On the other hand, if you're making the leap from another Web app development platform to J2EE, you have an entirely separate set of problems, not the least of which is coping with learning Java in general, and then all the capabilities of J2EE. You not only need to figure out how to apply your previous Web app experience in this new platform, but you need to deal with peculiarities and unique features enabled in J2EE.

The problem is, where do you begin? Do you start with a J2EE book? A JSP book? A servlet or EJB book? You'd better be careful. Almost all of these will presume you already understand Java. This may have made sense previously in that most writers were speaking to the vast army of Java developers making the move to the server-side, or enterprise, platform. But it leaves many newcomers to both Java and J2EE unable to start with those books.

Many introductory Java books, articles, and classes, on the other hand, presume that the reader is coming from a C or even C++ background (or no programming background at all). Again, that may have been a reasonable marketing decision in the early days of the transition to Java, but as more Web app developers with ASP (VBScript) or ColdFusion (CFML) make the transition, they're often hard-pressed to appreciate the analogies and references to how much better (or simply different) Java is than C and C++.

A particularly strong example of this is the almost paltry coverage of objects, object-oriented design, and object-oriented programming in many introductory texts. Often it's given just a chapter (and in some books, just a section), which does a terrible disservice to newcomers to Java, though it may have made sense speaking to C++ programmers.

Another dilemma is that many of the resources available presume that (1) all those new to Java want to learn about building Web clients, and (2) all those new to J2EE are experienced Java developers "moving up" but lacking Web app experience. It can be a frustrating experience for many.

So this column will also speak to new J2EE developers who have previous Web app development experience, helping them make the transition to the powerful and incredibly rich Java and J2EE platforms.

A J2EE Newcomer's Introductory Java Library

Still, there's no getting around the fact that the first step is to learn the language and platform. Some may suggest that you can create a JSP page without knowing Java, and that may be true in the most generous sense, but you won't get far beyond the most trivial examples without a solid grounding in the fundamentals of Java.

Let me take a moment and commend a few highly regarded introductory books: Ivor Horton's *Beginning Java 2*, Cay Horstmann and Gary Cornell's *Core Java 2*, Bruce Eckel's *Thinking in Java*, and one you may not yet have heard of, Jacquie Barker's *Beginning Java Objects*.

Beginning Java 2 gets nearly unanimous praise for its lucid introduction to Java. More than a third of the book (seven out of 20 chapters) focuses on client-side application development, but the first several chapters are an excellent introduction to core Java.

This of course leads nicely into the book of the same name, *Core Java 2*. Again, it's roundly applauded as a seminal work for its encyclopedic yet approachable coverage – and indeed worthy of the praise – but it, too, is heavily laden with client-side development chapters (four out of 12).

Yet another foundational book is *Thinking in Java (2nd ed.)*, a tour de force introduction to Java as not just another language but as a new way to design and code applications (with only one of the 15 chapters – appropriately late in the book – covering applet development).

I highly recommend his *Hands-on Java* CD-ROM as well, which parallels the book and offers a multimedia version of his seminar of the same name. If you can get a portable MP3 player (there are some that will play straight from the CD, or you can download files to it), you can listen to it on the train, during a workout, in the car, or wherever, and hear it over and over. It's a wonderful way to get grounded in the fundamentals (and some of the dark corners) of Java.

Finally, the most recent of these is the one that I feel does the best job of filling in the gaps left by the others in their coverage of objects and object-oriented programming: *Beginning Java Objects*. While some have said it's not the first book they'd recommend for newcomers to Java, I wonder if they're bringing a bias of already understanding objects. I found Barker's book wonderfully refreshing and straightforward on that subject, considering the alternatives, while also serving as an adequate introduction to the Java language.

Indeed, all these books are excellent resources for even the more experienced programmer. As Barker points out, it's easy to write Java code that doesn't truly leverage objects. It's a terribly ineffective way to do so, but entirely possible. I even recommend it to experienced Java developers (all but the most experienced or cynical).

Of course, these aren't the only Java introductory books (not even a fraction of the total number), nor are they necessarily the best for everyone. This is just one person's opinion (backed by similarly favorable reviews and awards on many sites). Before I leave the subject of getting a good head start on Java, there are several Web sites you can refer to (perhaps the most prominent being http://java.sun.com), and magazines (including the one you hold in your hand) as well. Be sure to ask your cohorts for their recommendations. There are too many to list.

AUTHOR BIO
Charles Arehart is a 20-year IT veteran with experience spanning a range of technologies, including very large-scale database systems. For the past four years, he's been an active trainer, writer, and consultant in enterprise Web application development.

Getting Started with J2EE: Resources

Now that you have a few resources to get you off the ground with Java, or if you're already comfortable with the core language, the next step is to become familiar (even intimate) with the J2EE platform. Again, there are several resources for this, including books, Web sites, and of course the magazine you're reading, among others.

As mentioned before, http://java.sun.com may be the best place to get started, specifically http://java.sun.com/j2ee/. This incredibly rich and deep site has a seeming never-ending supply of resources for the J2EE developer, including the J2EE Tutorial, Blueprints, case studies, tools, and white papers.

Other prominent portal (information) sites include www.theserverside.com, www.jguru.com, www.jspinsider.com, and more. Even if you're using a competing Java application server, you should also consider sites such as www.ibm.com/developerworks/ and http://.developer.bea.com. These offer documentation, tools, newsgroups, source code, articles, user groups, and many other resources.

As for books, there are many popular ones from a variety of standpoints, including:

- *Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition* (the printed version of the J2EE Blueprints)
- *Core Servlets and JavaServer Pages* (Marty Hall)
- *Professional Java Server Programming J2EE Edition* (Wrox Press)

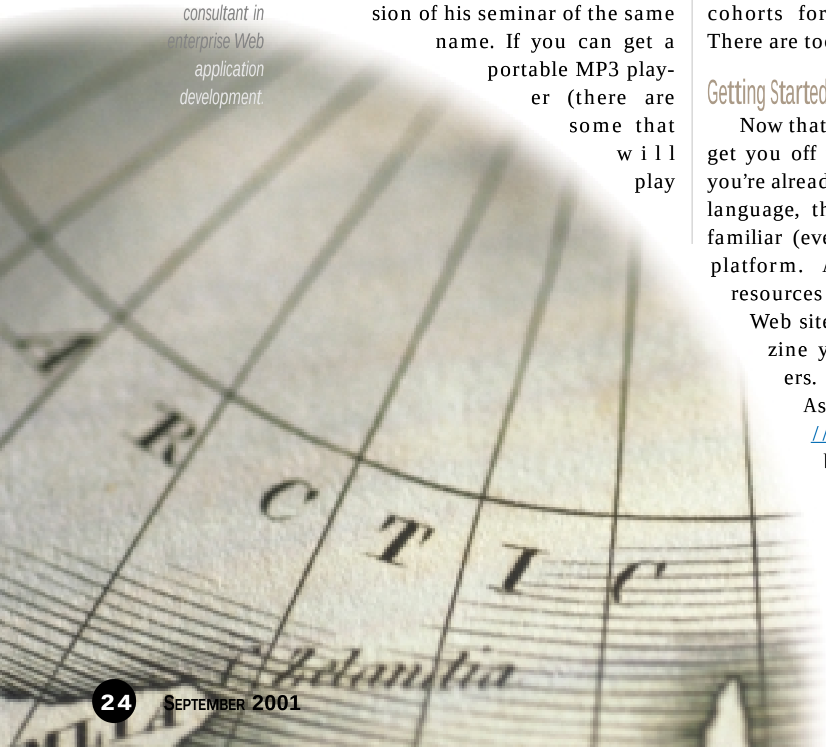
There are many more, including books on EJBs, servlets, and JSPs. Ask your colleagues, or visit portal sites and book review sites for more opinions. Just be aware that almost all of these books presume you have Java experience. And even then they may presume you don't have prior Web application development experience, so be prepared for some review-level material on those aspects (such as, "A form can have 5 kinds of input controls...").

Finally, as you embark on your own journey into the world of Java and J2EE, you may also need to learn about such topics as UML (and object-oriented design in general), patterns (both general design and J2EE), and many more related topics that separate the newcomer from the professional.

Summary

Clearly, for those making the transition from Web application development in ASP/CF/PHP, and others, there's a need for a book that meets both the foundational needs of learning Java (without too much client-side focus) while also offering foundational J2EE material (without presuming Java experience or repeating already understood Web app development fundamentals). I hope someone out there is listening and realizes there's a market. I've considered it myself, but I'm busy enough for now! I hope you'll join me in future installments of **Journeyman J2EE**.

carehart@systemmanage.com



JAVA RULES IN J2EE

written by Steve Ross-Talbot

Over the last two decades rules have become an increasingly important part of the information technology landscape. In fact, *deductive* rules have been applied to databases since the inception of SQL and form the basis of policy management and decision making at most corporations. However, the rules are changing. A newer *reactive* rules solution based on events, conditions, and action (ECA) has come to the foreground.

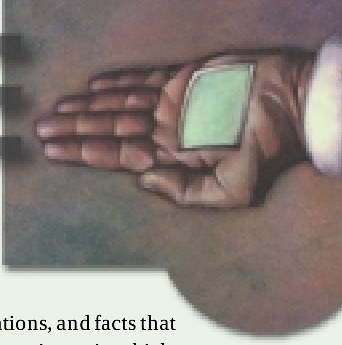
In addition, the rise of the Java runtime and development environment from a niche solution to prime time is also changing the rules of the information technology landscape. With Java as the core technology for embedded and enterprise applications, rules are an ideal means for businesses to harness the power of the language for a competitive advantage.

This article examines both types of rules, deductive and reactive, shows when and how to apply them, and explains how rules can work together to provide a more effective semantic integration solution. It also examines the different uses of rules and applies rule technology to J2ME for desktop applications and J2EE for enterprise applications.

This article looks at the use of rules for managing personal workflow, business transactions, and Web content in J2ME, J2EE, and JSP environments, respectively. It also covers the application of rules to B2B, customer relationship management (CRM), and supply chain management (SCM) applications.

An Overview of Rules

According to the authoritative business report, GUIDE Business Rule Project (1995), a business rule is “a statement that



defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business."

As rules have become an increasingly important aspect of the management of the IT landscape, this article examines the need for rules using a financial services example and looking at the different rules and how they can be applied to business-critical events in financial services. In the past, application builders have embedded the fundamental business-processing rules directly into application code. This led to an explosion of new applications when applications became too unwieldy and inflexible to meet business needs. It also led to an increase in maintenance as applications were modified to incorporate the necessary changes. This is more apparent in financial services where regulatory requirements and the personalization necessary to keep or gain market share are critical business factors.

Many systems have been written and rewritten trying to keep up with the changes. Business requirements are changing faster than applications can be created and/or modified. Rules offer a way of encapsulating the business semantics and promoting them to the surface in the same way that databases enabled the separation of data from an application. The first attempts at doing this date back to E.F. Codd, the founder of relational databases.

Capturing the semantics, or gaining an understanding of what is really going on in terms business people can understand, is one of the challenges to rules-based solutions. Rules are not only for computers, people also use them for heuristic decision making. Many a trader monitors events from a desk, taking in news stories, price changes, trading behavior, current position, and even credit exposure information, then uses it to decide whether to buy or sell a particular financial instrument. The layman may think the decision is based on a "thumb-in-the-air" approach, but it's not. Experience and judgment are valuable assets in today's trading environments. No rule system can replace this, but it should be able to support it.

Managing events, bringing them together, and actively supporting the people who make the decisions is the primary goal of rules-based solutions. Rules enable us to capture the business semantics embedded in applications to support the decision-making processes in an organization. Businesses need to remain flexible so that the IT infrastructure can support complex real-time decision making.

Two Kinds of Rules — Deductive and Reactive

The deductive rule approach, which uses forward and backward reasoning, has been around for the last 20 years and was born from the artificial intelligence (AI) community. Deduction (or inferencing) is a means of deducing facts from an existing knowledge base. Unification is a technique that's commonly used to infer or deduce facts. A typical example follows:

- ***Rule:*** A preferred customer is one who has spent more than $10,000 in the last month.
- ***Fact:*** John Doe spent $11,000 last month.

From the fact and the rule it's possible to deduce that John Doe is indeed a preferred customer. Another, more complex goal-oriented example is shown below:

- ***Rule:*** *x* is the grandfather of *z*.
- ***Implication:*** *x* is the father of *y* and *y* is the father of *z*.
- ***Fact:*** Joe is the father of John.
- ***Fact:*** John is the father of Fred.

This form of reasoning is based on rules, implications, and facts. It's goal-oriented, allowing us to deduce that "Joe is the grandfather of Fred" by using unification over the rules, implications, and facts that exist in the knowledge base. This technique is one in which you attempt to prove conclusions. It allows reasoning to take place over unbound variables, which the ECA rules aren't well suited for.

The Rete algorithm is one of the best-known algorithms for doing this. It builds up a treelike structure for the knowledge base and deduces facts efficiently (or proves conclusions) by evaluating only what is necessary. This saves enormous processing time and memory by targeting only those parts of the "tree" that need to be reevaluated against a conclusion. Typical systems based on this technique are Nisus Inc.'s Nisus rules engine, iLOG's JRules, and Brokat's Advisor/J.

Reactive or ECA rules have been around for a shorter time; however, their heritage is in the definition of Sequel2 (from the article "Implementation of a Structured English Query Language," by Morton M. Astrahan and Donald D. Chamberlin) and their triggers are in programming language exception handling as far back as PL/1. In the 1990s, ECA rules were a fundamental component in active databases, and more recently in the semantic integration that underpins B2B and Web services.

ECA rules are well suited to event-centric problems, which deal with change and how to manage it. In a database trigger, they might look for a deletion of a tuple from a department table, and as a consequence, delete all employees. In a more complex active database example they might look for the deletion of the same department tuple, but seek guidance on whether this is meant to be a department name change, transfer, or a downsizing based on departmental pruning. In the simple trigger case, the semantics of the deletion can't be modeled, but they can in the active database example.

In the late 1990s, ECA rules started to leave the confines of the database transaction world and exist as agents of system-to-system workflow. In this context they've been used as rule agents to police business change. The rules help consolidate business events into business transactions and monitor business events to enable an enterprise to react in a more ad-hoc fashion, capturing some of the business heuristics that underpin modern business decision making.

An example of a business transaction in a financial services setting is shown below:

```
ON NewTrade EVENT FOLLOWED BY A SettledTrade EVENT WITHIN 5 MINS

CONDITION NewTrade.id MATCHES SettledTrade.id AND
NewTrade.amount EQUALS SettledTrade.amount
ACTION PUBLISH "Trade settled fully"

CONDITION NewTrade.id MATCHES SettledTrade.id AND
                    NewTrade.amount GREATER SettledTrade.amount
ACTION PUBLISH "Partial settlement"

ON TimeOutException EVENT OF A SettledTrade
ACTION PUBLISH "Trade failed to settle in time"
```

In this example the rule is composed of a target event, the NewTrade, and followed by an event, the SettledTrade. The rule states that when a NewTrade and a matching SettledTrade are found and the amounts match for the two events, the trade is said to be fully settled. If, however, the amounts don't match, the trade is said to be partially settled. And if no matching SettledTrade event is found within 5 minutes, the trade is considered to have failed to settle in time. This is a fairly typical business-transaction monitor process that matches orders to confirmations, and models time relationships to do so. This same example is shown graphically in Figure 1.

```
ON NewTrade EVENT FOLLOWED BY AN ExposureChange EVENT WITHIN 30
MINS
CONDITION NewTrade.longName EQUALS "Barclays Bank" AND
ExposureChange.country.longName EQUALS "Guatemala" AND
ExposureChange.limit/0.9 LESSTHAN ExposureChange.limit-
  ExposureChange.remaining
ACTION CALL RiskMgmtSystem.Embargo("Mexico")
```

A monitoring rule with flexible teeth is also shown. This rule looks for NewTrade events followed by ExposureChange events from a risk system. If the trade is for "Barclays Bank," the country of origin is "Guatemala," and only 10% of the allowable limit is left, all trades to "Mexico" are embargoed. Why would we want to do this? Simply, it's the management of the unpredictable that is one of the many benefits of ECA rules. It's the ability to juxtapose situations (in the form of events and conditions) and act on them (in the form of actions) that's critical to the ability of a business to react. This is what provides a competitive edge to the decision making within an organization. It allows the systems to support the users rather than force them to change their habits to work with the systems. An example of an ECA system that provides the basis for reactive rule specification in RuleML is SpiritSoft's SpiritIntellect.

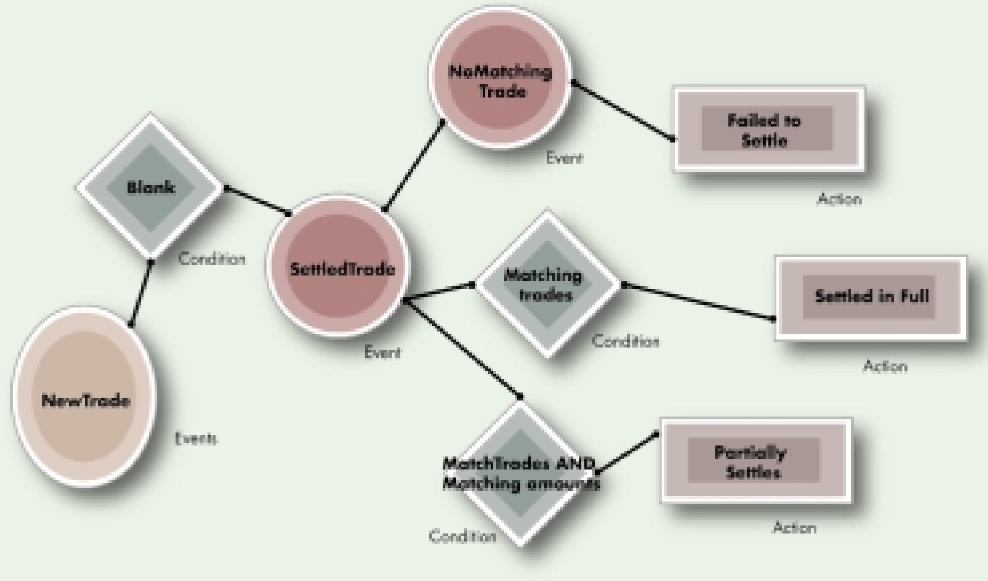


FIGURE 1 An ECA rule

Rules and Java

Java, especially with J2EE, has breathed new life into rules technology. Since last year two new Java Community Process (JCP) initiatives have started, JSR 87 and JSR 94, both of which have rules as their central component. As the shift has moved to Web services over a semantic Web, we've seen the rise of RuleML from W3C, the most mature of the rules initiatives. The fundamental reason for this is a drive to capture business semantics and to increase the scope for personalization. In both cases the initiatives are predicated on a desire to achieve a more semantic form of business integration in a B2B Web services context.

The role of Java should not be underestimated. The ability to call Java code from within rules has led to a more fundamental integration story for rules technology. The role played by technologies such as EJB application servers, JSP, the JMS API, and XML have all contributed to the rise of rule-based technology as businesses look to differentiate their offerings. This has led the way in promoting personalization of offerings to customers through EJB and JSP technology, which allows users to be in control of policy decisions as they relate to applications at the user interface and business workflow levels.

Since rules in a Java environment have been the focus for the last 12 months, we'll look at how rules can be used in different Java platforms with different Java technologies. Rules engines, along with many other systems, have suffered from integration problems with their surrounding environment. Java has made it possible for rules to dynamically call out to Java methods to incorporate flexibility into a system that has been lacking. To enable smoother and more cost-effective integration, the Java programming model and supporting environment makes it possible to introspect Java objects and classes while a program is running. This achieves a more dynamic integration between rules and Java, particularly if the rules engine or framework is written in Java.

Rules and J2ME

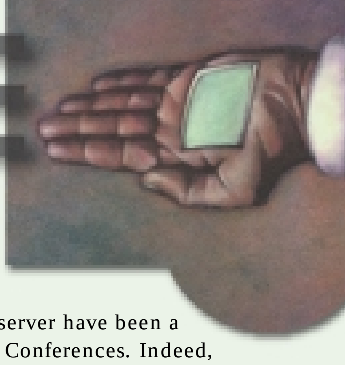
In a world where we're always connected, we need to ensure predictable results. By definition, a mobile environment inherently includes variability of the network and has limited bandwidth. New paradigms are required for transacting business that deals with decoupled, shareable business-transaction contexts and enables adaptable applications to be constructed.

PocketWorkflow allows the user to configure mobile applications to be both adaptable – graying out options as changes to the effective Quality of Service (QoS) occurs – and transactional – by providing on-the-fly business transaction delegation. To do this, PocketWorkflow uses ECA rules to provide a framework for adaptability and for business transaction delegation, and uses deductive rules to deduce overall context while sifting through e-mail, voice, and other information sources.

The marriage of deductive and ECA rules in this domain shows that each plays an important part in the overall solution. Taking advantage of the data-centric model of deductive rules applied over a knowledge base, it's possible and cost-effective to deduce short-term context, which can act as a filter to increase relevance to mobile devices and ensure that they're not swamped with irrelevant information. Leveraging the event-centric model of ECA rules, it's possible to provide user-centric, flexible PocketWorkflow to manage applications' adaptability to a changing environment and the business transactions that they participate in.

Rules and J2EE

In the world of J2EE, rules have been increasingly used to personalize applications. It has addressed two solution spaces to date, and with the advent of Web services, a new set of requirements arises.



Existing requirements relate to validation and verification of data entry (e.g., orders) and the management of integrity constraints within EJB application servers. Newer, Web services–related requirements need to be met to ensure declarative privacy and digital rights management. Both of these are topics that have received much attention at W3C under Web services and the semantic Web.

An architectural view of J2EE technology and rules is shown in Figure 2.

This architecture positions rules with respect to the requirements they need to meet and the technologies as they are today.

Rules and JSP

Rules have a natural role to play in JSP-based solutions. Being closer to the front of the application, they play an important role managing validation and verification issues as they relate to order entry and what you should be seeing. Validation and verification of order entry is best done nearer the user to prevent spurious server-related bottlenecks. Such validation and verification is best served if it can be flexible and personalized. Using a rule-based solution enables this to be done in a cost-effective manner.

Ensuring the integrity of a form is something that deductive rules systems have long been used to. Once the form is filled in, ECA rules play a role in managing the business transaction. Further requirements that underpin what you might see can be driven through deductive rules and their ability to police digital right management and infer what you can see. In this role they play a key part in ensuring that the dynamically generated HTML pages are the correct ones, and that you see what you're entitled to see and nothing else.

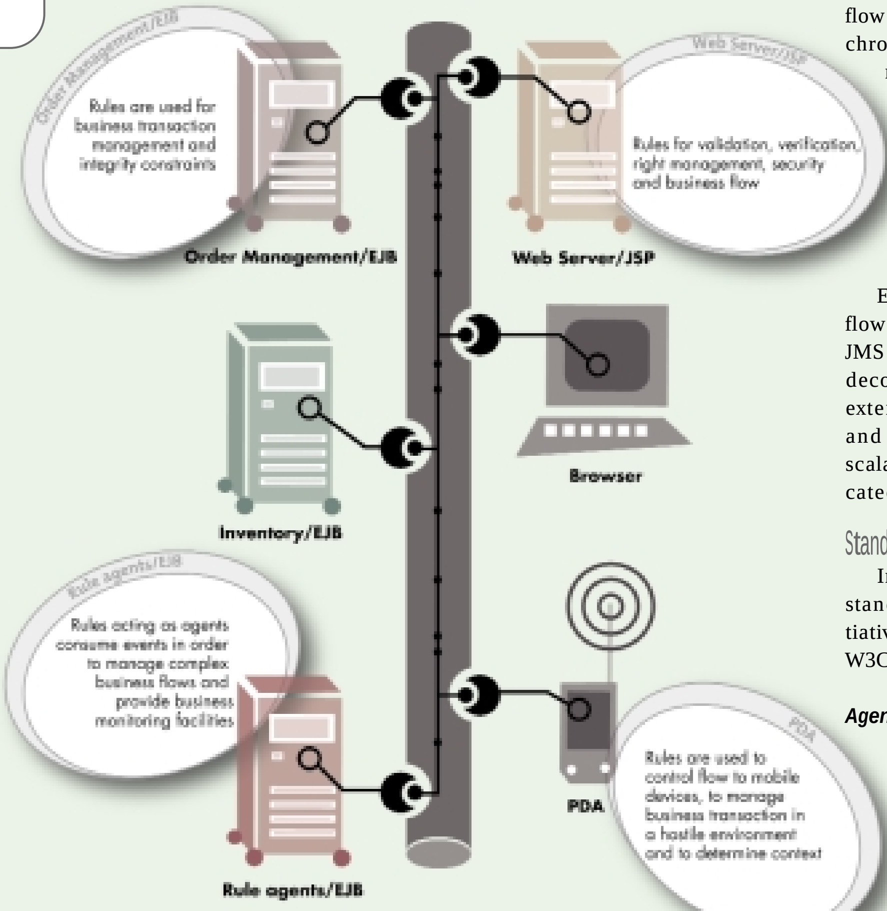

Order Management/EJB
Rules are used for business transaction management and integrity constraints
Web Server/JSP
Rules for validation, verification, right management, security and business flow
Order Management/EJB
Web Server/JSP
Browser
Inventory/EJB
Rule agents/EJB
Rules acting as agents consume events in order to manage complex business flows and provide business monitoring facilities
PDA
Rules are used to control flow to mobile devices, to manage business transaction in a hostile environment and to determine context
PDA
Rule agents/EJB

FIGURE 2 J2EE and where rules fit

Rules and EJB

Rules applied to an application server have been a hot topic for at least two JavaOne Conferences. Indeed, many users of EJB technology are using rules to add flexible integrity constraints through a specialized form of transaction-coupled ECA rules. Deductive rules are used to manage policy issues as they relate to what you can and cannot see, and the ECA rules further augment this by enabling the user of the technology to express how things should happen in a personalized context. In this way, a deductive rule might allow an application to understand who is a platinum, gold, silver, or bronze customer based on their service-level agreement, and supply this information to an ECA rule that determines which internal components are to be activated (e.g., maybe no credit checking is done for a platinum customer, maybe limited credit checking is done for gold, and so on).

These policy decisions make the difference between services that are okay and those that are responsive to customer demands. They enable applications to be built that are flexible enough to take advantage of changing business models and therefore meet changing business requirements.

Rules and JMS

JMS is not an obvious place for rules to play a part. If we step back and look at workflow in the early 1990s, many of the workflow products that had their genesis at that time used asynchronous messaging techniques to act as a distribution channel for workflow events. In the early days of IBM's Flowmark (now MQWorkflow), an internal asynchronous messaging bus did all that MQSeries did. So JMS and event management as it relates to managing business events cries out for event-centric rule mechanisms to describe and manage the flow of events as they relate to business transactions and business processes.

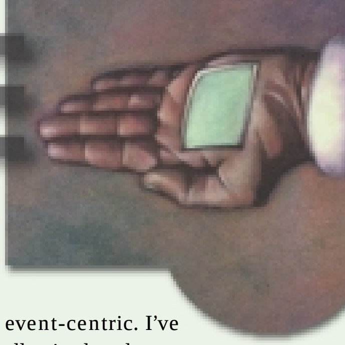
Event-centric rules are a natural way to express process flow between applications, services, and components. In a JMS context they provide a way to express process flow that's decoupled from the underlying application services. They extend and complement what can be achieved within a JSP and EJB environment by providing a distributed and highly scalable way to manage business transactions and sophisticated flexible monitoring facilities.

Standards and Rules

In this section we look at rules in the context of a number of standards initiatives. We cover JSR87 and the agent services initiative, JSR94 and the Java rules initiative, and the work in W3C.org on RuleML and the semantic Web.

Agent Services and Rules

Like many of the JSR initiatives, JSR87 defines a set of objects and service interfaces. In the case of JSR87, these are to support the deployment and operation of autonomous communicative agents based on the Foundation for Intelligent Physical Agents (FIPA) abstract architecture. This, in turn, is influenced by the work of Darpa Agent Modeling Language (DAML), which has been a major part of the semantic Web initiative at W3C.



Agents that can be discovered and are flexible and personal have given rise to rules-based flexibility. It's in this field that JSR87 is important and in particular its relationship to FIPA, DAML, the semantic Web, and RuleML. This is discussed later in the article.

JSR94 – Rules and Enterprise Java

JSR94 defines a Java runtime API for rules engines. The API prescribes a set of fundamental rule-engine operations. This set is based on the assumption that most clients will need to execute a basic multistep rule-engine cycle that consists of parsing rules, adding objects to an engine, firing rules, and getting resultant objects from the engine. It doesn't separate rule types (deductive and ECA), assuming some variant of a Rete model of rule computation. It also doesn't look at defining a common rule language. Rather it looks at a partial mapping to a Rete-based rule framework to define a common parsing API for rule sets so that they may be shared.

The work of W3C on RuleML goes substantially further. However, JSR94 is the only JSR that's been initiated to deal with rules, albeit a subset of the rule space. It's interesting to note that JSR94 refers to business rules from IBM and was the work of Benjamin Grosof, one of the key players in RuleML.

RuleML and Java-Based Rules

Rules for the Web have become a mainstream topic and have been identified as a design issue for the semantic Web since its inception. RuleML is an open initiative that seeks to work toward an XML-based markup language that permits Web-based rule storage, interchange, retrieval, and firing/application. It's the only rule standards initiative to deal with deductive and reactive rules within the same framework. It openly seeks to leverage reactive and deductive power to provide a more meaningful interchange of rules-based technology, which is necessary to power the next generation of Web services and semantic Web applications.

Being a W3C initiative provides a language-neutral approach that's likely to lead to wider ranging results than one that's tied to Java. The incorporation of reactive rules through an ECA-like rule definition provides this initiative with a wider canvas than just deductive-based solutions. Having both types of rules under the same umbrella may lead to more interesting and pervasive solutions.

Rules for the Semantic Web and Web Services

Rules are a fundamental currency in managing Web services. For the first time Web services have dealt with the flow of control between components (or services) that's required to achieve some goal. In a business context this maps to a business transaction. This flow can be defined by a choreographed set of events, and the control and flexibility can be expressed using ECA rules and enhanced using deductive rules. This enables service choreography to take advantage of the natural event-centric model of ECA rules and the data-centric model of deductive rules to declaratively express conditions (e.g., are you a preferred customer?).

The semantic Web work of W3C is highly relevant to all of this. Indeed, Tim Berners-Lee, director of the W3C, has gone on record saying, "Web services are an actualization of the semantic Web." To this end, much of the work on DAML, RDF, and RuleML has a high degree of relevance to Web services, and it's likely that Web services will be the first user of the technology that underpins the semantic Web. It's our belief that RuleML and RDF-like semantic decoration will enable better service discovery and personalization to take place. This in turn will lead to a new generation of Web-based application solutions.

Summary

What I've shown is that there are fundamentally two types of application areas in which rules have a major role: data-centric and event-centric. I've illustrated that deductive rules are well suited to data-centric problems and ECA rules to event-centric problems. I've examined the different deployments of rules from J2ME to J2EE, and looked more closely at how the different rule models add value. I've given a précis of current standards-based work, looking at the JSR initiatives for agents and rules, introducing RuleML, and tying them together around Web services and the semantic Web.

The rules that we discussed relate to personalization based on standard license agreements with services that can impact what you see, what you do, and how you do it. These are all critical to the success of any B2B solution, SCM offering, or ERP initiative. All these business domains require value-added flexibility to enable their systems to meet the needs of the enterprise and the multiplicity of roles within it.

References

1. Snell, J. (2001). "The Web Services Insider, Part 2: A Summary of the W3C Web Services Workshop." April. www-106.ibm.com/developerworks/webservices/library/ws-ref2/?dwzone=webservices
2. GUIDE Business Rule Project, Final Report, Nov 6, 1995.
3. Tombros, D. (1999). An Event- and Repository-Based Component Framework for Workflow System Architecture. PhD Thesis, University of Zurich, November.
4. Astrahan, M.M., and Chamberlin, D.D. (1975). "Implementation of a Structured English Query Language." Comm. ACM 18:10, pp 580–587.
5. Forgy, C.L. (1982). "Rete: A Fast Algorithm for the Many Pattern/Many Object." Pattern Match Problem. *Artificial Intelligence*. pp 17–37.
6. Ross-Talbot, S., Brown, G., et al. (1998). "Building Globally Adaptive Systems." ObjectExpo, Europe. November.
7. Ross-Talbot, S. (1998). *Object Databases in Practice*. Prentice Hall.
8. *RuleML:* www.dfki.uni-kl.de/ruleml/
9. Codd, E.F. (1979). "Extending the Database Relational Model to Capture More Meaning." ACM Transactions on Database Systems 4. pp 397–434.
10. Fritschi, H., Gatziu, S., and Dittrich, K.R. (1997). "FRAMBOISE: An Approach to Construct Active Database Mechanisms." Technical Report 97.04, Department of Computer Science, University of Zurich. ftp://ftp.ifi.unizh.ch/pub/techreports/TR-97/ifi-97.04.ps.gz
11. Hsu, M., Ladin, R., and McCarthy, D. (1988). "An Execution Model for Active Data Base Management Systems." Third International Conference on Data and Knowledge Bases, June.
12. *Business Rules for Electronic Commerce:* www.research.ibm.com/rules/home.html

AUTHOR BIO

Steve Ross-Talbot is CTO and principal founder of SpiritSoft. He holds the position of honorary research fellow at Napier University, Edinburgh, Scotland, having published work ranging from query optimization to advanced ECA-rule architectures.

steve.ross-talbot@spirit-soft.com

Recognizing and Eliminating Errors in Multithreaded Java

How to handle deadlocks and data races

WRITTEN BY
MARK DYKSTRA

Threads are essential to building high-performance, scalable applications, and are especially critical in server-side Java applications. Writing multithreaded Java is complicated, even for experienced developers.

Errors in multithreaded programs may not be easy to reproduce. The program may deadlock or encounter other thread-related errors under only very specific circumstances, or may behave differently when running different VMs.

If you use multithreading in your client- or server-side Java, you should seriously consider a detection solution for the most common problems with threaded programming, including:

- Deadlocks
- Potential Deadlocks
- Data Races

Deadlocks

A deadlock is a situation where threads are blocked because one or both are waiting for access to a resource that will not be freed. The application can never terminate because the threads are blocked indefinitely.

This behavior results from improper use of the synchronized keyword to manage thread interaction with specific objects. The synchronized keyword ensures that only one thread is permitted to execute a given block of code at a time. A thread must therefore have exclusive access to the class or variable before it can proceed. When it accesses the object, the thread locks the object, and the lock causes other threads that want to access that object to be blocked until the first thread releases the lock.

Since this is the case, by using the synchronized keyword you can easily be caught in a situation where two threads are waiting for each other to do something.

A classic example for a deadlock situation is shown in Listing 1. Now consider this situation:

- One thread (Thread A) calls method1()
- It then synchronizes on lock_1, but may be preempted at that point.
- The preemption allows another thread (Thread B) to execute.
- Thread B calls method2().
- It then acquires lock_2, and moves on to acquire lock_1, but can't because Thread A has lock_1.
- Thread B is now blocked, waiting for lock_1 to become available.
- Thread A can now resume, and tries to acquire lock_2. It can't because Thread B has acquired it already.
- Thread A and Thread B are blocked. The program deadlocks.

Of course, most deadlocks won't be quite so obvious simply from reading the source code, especially if you have a large multithreaded program. A good thread analysis tool, like Sitraka's JProbe Threadalyzer, finds deadlocks and points out their location in the source code so that you can fix them.

Potential Deadlocks

Potential deadlocks are caused by problematic coding styles that might not cause a deadlock in every test execution. For that reason, they are perhaps more dangerous than deadlocks, as they may remain hidden until after the application is deployed. We'll discuss two types of potential deadlocks: Lock Order and Hold While Waiting.

Lock Order

Lock order violations can occur when concurrent threads need to hold two locks at the same time. The potential for deadlock develops when one thread holds a lock needed by another. Consider the situation where Threads A and B both need to hold locks 1 and 2 at the same time.

It is possible that events could unfold as follows:

- Thread A acquires lock_1.
- Thread A is preempted and the VM scheduler switches to Thread B.
- Thread B acquires lock_2.
- Thread B is preempted and the VM scheduler switches to Thread A.
- Thread A attempts to acquire lock_2 but is blocked because lock_2 is held by Thread B.
- The scheduler switches to Thread B.
- Thread B attempts to acquire lock_1 but is blocked because lock_1 is held by Thread A.
- Threads A and B are now deadlocked.

It's important to note that this deadlock might not occur in some situations. The VM scheduler might allow one of the threads to acquire lock_1 and lock_2 in succession, without preempting the thread. In such a case, regular deadlock detection would not report it.

A fully featured thread analysis tool would track the order in which locks are acquired, and warn of any problematic lock ordering. A lock order analysis feature should issue warnings whenever the VM scheduler might deadlock, while deadlock detection should report only actual deadlocks.

Hold While Waiting

Another type of potential deadlock occurs when a thread holds a lock while waiting for notification from another thread. Consider the example shown in Listing 2.

This code is problematic in that Consumer can hold the lock on the queue, denying Producer the access it needs. This can occur even if Consumer is waiting for Producer to send notification that another item has been added to the queue. Since Producer can't add items to the queue, and Consumer is waiting on Producer for new items to process, the program is effectively deadlocked.

Locks held while waiting are only potential deadlocks because events could transpire in such a way that the notifying thread does not need the lock held by the waiting thread. However, such programming practice is risky unless you are absolutely sure that the notifying thread will never need the lock. Locks held while waiting can also cause cascading stalls, where one thread idles while holding a lock needed by another thread, which in turn holds a lock needed by yet another thread, and so on.

To correct the previous example, modify the Consumer class by moving wait() outside of synchronized(), as follows:

```
public class Consumer
 {
  synchronized void consume()
   {
    while (! done) {
          wait();
          synchronized(queueLock_) {
            removeItemFromQueue
            AndProcessIt();
   }
  }
 }
}
```

Data Races

A data race results from a lack of synchronization or the improper use of synchronization when accessing shared resources such as variables. Data races occur when the developer fails to specify which thread has access to a variable at a given time. In such a case, whichever thread wins the race gets access to the data, with unpredictable results.

Because threads can be preempted at any time, you can't safely assume that a thread executing at start-up will have accessed the data it needs before other threads begin to run. As well, the order in which threads are executed may differ from one VM to the next, making it impossible to determine a standard succession of events.

“Writing multithreaded Java is complicated, even for experienced developers”

Sometimes, data races may be insignificant in the outcome of the program, but more often than not they can lead to unexpected results that are hard to debug. In short, data races are concurrency problems waiting to rear their ugly heads. A good thread analysis tool will identify any data race it encounters while executing your program, and flag it for you to fix.

A Benign Data Race

Not all race conditions are errors. Consider the example in Listing 3.

Assuming that getHouse() returns the same house to both threads, you might conclude that a race condition is developing because the BrickLayer is reading from House.foundationReady_ and the FoundationPourer is writing to House.foundationReady_.

However, the Java VM specification dictates that Boolean values are read and written atomically, meaning that the VM can't interrupt a thread in the middle of a read or write, and that once the value has been changed, it's never changed back. This is a benign data race, and the code is safe.

A Malignant Data Race

Now, consider the following scenario in Listing 4.

What happens if a wife and husband simultaneously attempt to deposit money to a joint account, from two different banking machines? Let's call them Alice and Bob. At the beginning of our scenario, their joint account has $100.

Alice deposits $25. Her banking machine starts to execute deposit(). It gets the current balance ($100), and stores that in a temporary local variable. It then adds $25 to that balance, and the temporary variable holds $125. Then, before it can call setBalance(), the thread scheduler interrupts her thread.

Bob deposits $50. While Alice's thread is still in limbo, his thread starts to execute deposit(). The getBalance() returns $100 (remember, Alice's thread hasn't written the updated balance yet), and the thread adds $50 to obtain a value (in its temporary local variable) of $150. Then, before it can call setBalance(), Bob's thread is interrupted.

Alice's thread now resumes, and writes its temporary local variable's contents ($125) to the balance. The banking machine informs Alice that the transaction is complete. Bob's thread resumes, and writes the contents of its temporary local variable ($150) to the balance. The banking machine informs Bob that the transaction is complete.

Net effect? The system has lost Alice's deposit.

Your first instinct might be to protect the Account.balance_ field by making getBalance() and setBalance() synchronized methods. This will not solve the problem. The synchronized keyword will ensure that only one thread can execute getBalance() or setBalance() at a time, but that won't prevent one thread from modifying the balance of an account while the other is halfway through a deposit.

How to Fix the Race

The key to successful use of the synchronized keyword is to realize that you need to protect entire transactions from interference by other threads, not just single points of data access.

In our example, the developer must ensure that once a thread has obtained the current balance no other thread can alter that balance until the first thread has finished using that value. This can be accomplished by making deposit() and withdraw() synchronized methods.

AUTHOR BIO
Mark Dykstra is Web content manager at Sitraka and has been working as a Web developer and technical writer for the past five years.

The Synchronized Keyword

Deadlocks, potential deadlocks, and data races are common multithreading errors made by developers of all levels of experience. The correct use of the synchronized keyword is essential to writing scalable, multithreaded Java code. A good thread analysis tool like Sitraka's JProbe Threadalyzer makes error detection much less laborious and is particularly valuable for finding problems that might not arise in every test execution.

This article is meant to be an introduction to the most common Java multithreading development errors. For more information on concurrent programming, refer to the References. Christian Jaekl was particularly helpful and I am grateful for his support and advice.

References

1. Jaekl, C. (1996). "Event-Predicate Detection in the Debugging of Distributed Applications," University of Waterloo. www.sitraka.com/jaekl96eventpredicate.pdf
2. Lea, D. (1999). *Concurrent Programming in Java: Design Principles and Patterns*, 2nd Edition, The Java Series.
3. Oaks, S., and Wong, H. (1999). *Java Threads*, 2nd Edition, O'Reilly.
4. Hartleys, S. (1998). *Concurrent Programming: The Java Programming Language*, 1998. Oxford University Press.

mad@sitraka.com

Listing 1

```
class Deadlocker
{
  int field_1;    private Object lock_1 = new int[1];
  int field_2;    private Object lock_2 = new int[1];
  public void method1( int value )
     {       synchronized( lock_1 )
      {       synchronized( lock_2 )
        {
          field_1 = 0;
          field_2 = 0;
        }
      }
    }
  public void method2( int value )
    {       synchronized( lock_2 )
      {       synchronized( lock_1 )
        {
          field_1 = 0;
          field_2 = 0;
        }
      }
    }
}
```

Listing 2

```
 public class queue {
  static java.lang.Object queueLock_;
  Producer                producer_;
  Consumer                consumer_;

  public class Producer
  {
       void produce()
       {
           while (! done) {
             synchronized (queueLock_) {
                  produceItemAndAddItToQueue();
                  synchronized (consumer_) {
                          consumer_.notify();

                  }
             }
           }
  }

    public class Consumer
     {
       consume()
       {
           while (! done) {
                      synchronized (queueLock_) {
                           synchronized (consumer_) {
                                consumer_.wait();
           }
                                 removeItemFromQueueAndProcessIt();
                      }
                 }
        }
    }
 }
```

Listing 3

```
 public class House {
  public volatile boolean foundationReady_ = false;
                 }

public class FoundationPourer extends Thread {
  public void run() {
    House a = getHouse();

// lay the foundation...

    a.foundationReady_ = true;
                }
}

public class BrickLayer extends Thread {
  public void run() {
    House a = getHouse();

//Wait until the foundation is ready
//NB:  This is a "busy wait", and is the *WRONG* way to
//do this; we should use wait() and
//Object.notify() instead.

      while (!a.foundationReady_) {
                          try {
                            Thread.sleep(500);
                              }
                          catch (Exception e) {
                            System.err.println
                              ("Caught exception:  "+e);
                                              }
                                  }
                    }
}
```

Listing 4

```
 public class Account {
  private int balance_;   // amount of money in the account, in cents

  public int getBalance(void) {
    return balance_;
                              }
  public void setBalance(int setting) {
    balance_ = setting;
                                      }
}

public class CustomerInfo {
  private int   numAccounts_;
  private Account[] accounts_;

  public void withdraw(int accountNumber, int amount)
          {
            int temp = accounts_[accountNumber].getBalance();
            temp = temp - amount;
            accounts_[accountNumber].setBalance(temp);
          }

  public void deposit(int accountNumber, int amount)
          {
            int temp = accounts_[accountNumber].getBalance();
            temp = temp + amount;
            accounts_[accountNumber].setBalance(temp);
          }
}
```

Download the Code!
www.JavaDevelopersJournal.com

J2ME | J2SE | J2EE | Home

mycgiserver

Hosted by netarray

mycgiserver
Web: www.mycgiserver.com
E-mail: general@mycgiserver.com

info

REVIEWED BY JASON BELL
jason@jflight.net

Sometimes finding hosting for your well-crafted pieces of code can be more work than the coding itself. Locating a service that does it free of charge is a real challenge; however, www.mycgiserver.com is a service that meets both criteria. The site started life as a CGI server that could run user's Perl scripts, PHP, and Java servlets, but in November 2000 they made the decision to concentrate on Java deployment.

There are other providers of free server space to run your Java Web applications, such as iSavvix (www.isavvix.com) or WebApp Cabaret (www.webappcabaret.com), so it is advisable to check all of them and see which will fulfill your requirements.

Many free services on the Internet contain banner advertising to fund the service. The only banner ads on www.mycgiserver.com are on their own home page and in the member's area pages. At present there are no banner ads on the Web space viewed by the public.

The servlet container used, Caucho Resin, is an open-source servlet engine. It also handles the JSP and XTP calls; being a Jakarta Tomcat user for so long I'm looking forward to researching and using Resin for my local development. We're not limited to servlets, JSP is also available, and for the data hungry there's InstantDB and Hypersonic SQL on the site. Access to an SMTP server and an RMI registry is also available.

Registration is all done online. Once registration details have been processed and you have replied to your confirmation e-mail, 5MB of server space is allocated. There is the option to be e-mailed at regular intervals to be told of upgrades (EJB is coming soon, plus plans for domain and subdomain hosting).

The members area gives information on the amount of space you have used, a brief rundown of the mappings to different file types (servlets, JSP, XML, XTP, and so on), the all important FAQ, and a list of installed components covering XML parsing, e-mail, regular expressions, and cryptography. Also available within the member's area is a Java compiler (JDK 1.3), so if you upload a Java source file to your area you can compile without having to do it on a local machine first. There is an option to password areas of registered Web space with ".htaccess", and a "htpasswd" program is available online to generate the passwords.

From a development point of view, all servlets created are contained as a package. It's important to note because if you don't declare any package information, the servlets won't run. As everyone shares the servlet engine, you have to declare that the servlet belongs to you (see Listing 1).

When users run the servlet, they would visit www.mycgiserver.com/servlet/fictionusername.TestServlet.

Servlets can run from any directory, but you must remember to adjust the package name accordingly in order for the servlet to work. For people new to the servlet and JSP programming, there are sample files for you to look at within your personal server space. There is also a discussion board, so you can post questions or provide answers to those in need or distress. This makes up for the lack of formal documentation and the depth of information within the FAQs.

There are a couple of minor problems with the service, the first being you can't keep all your servlet classes in one .jar file. The other is the online documentation. What I would like to see are pages for setting up the add-on components, such as InstantDB and Xerces, and using the RMI registry. For example, I had to download InstantDB onto my development machine and figure out how it works, then transfer that knowledge to my server account. My concerns are for new Java programmers who want to develop and learn. Our aim should be to encourage not discourage.

I use the service as a test bed and alternative Web space, and is something I'm looking to develop more. For those who don't need the complexity of JRun or WebLogic, I would encourage you to look at www.mycgiserver.com. The only cost is your time and effort.

Listing 1 TestServlet.java

```
package fictionusername;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class TestServlet extends httpServlet {
 public void doGet(httpServletRequest req, HttpServletResponse
res)
   throws IOException, ServletException {

   // body of code continues...

 }
}
```

Download the Code!
www.JavaDevelopersJournal.com

Design Patterns for Optimizing the Performance of J2EE Applications

Stop reinventing the wheel

WRITTEN BY
VIJAY S. RAMACHANDRAN

With the proliferation of J2EE as the platform of choice for server-side applications, the need for sharing developers' experiences and the availability of reusable designs has become crucial.

In this article, we get to know some of the reusable designs that can be used to improve the performance of a J2EE application. For the benefit of those who are not familiar with design patterns, a brief description is given at the beginning before delving into the details. For further details on design patterns, see the reference section at the end of this article.

What Are Design Patterns?

As software developers design and build different applications, we come across the same or similar problem domains. This leads us to find a solution for the same/similar problem everytime, and we end up "reinventing the wheel" again and again. It would be helpful to have a repository that discusses such common problem domains and proven solutions. It would be much better if such a repository discussed the problem domains, along with a solution best suited to solve the problem on hand. Such a solution could be the result of the hands-on experience of the development community.

In the simplest terms, such a common solution is a design pattern and the repository or place of reference that contains such patterns is a design-pattern catalog. A design pattern prescribes a proven solution from experienced hands for a recurring design problem. Apart from describing the problem and prescribing the solution, the pattern will also explain the implementation issues involved and consequences, if any, of using the pattern. These solutions are generic in nature. They are described in the well-defined "Pattern Templates"; the most common one in use is the template defined by the "Gang of Four." The pattern templates usually have a name that offers a good idea as to what that pattern is all about, followed by where the pattern is applicable, the motivation, implementation issues, and other descriptions.

Use of such patterns makes the design of an application transparent. These patterns have been used successfully by developers in their respective work and hence the pros and cons of their use as well as implementation issues are known beforehand. All design patterns are reusable and can be adapted to a particular context; this gives developers flexibility. The use of design patterns related to J2EE applications offer the added advantage of providing solutions for J2EE platform technologies.

Performance-Optimizing Design Patterns For J2EE Applications

This article does not explain the patterns with their formal template and a full-blown code sample. The "J2EE Blueprints" link in the reference section will be the place of reference for those interested in such details. Instead we look into some recurring problem domains that have a tremendous effect on the performance of a J2EE application. We also look into a working solution for each of the problem domains that we discuss, along with some important points to note and small code samples wherever applicable.

Multitiered J2EE applications consist of components communicating across tiers to access/change data. This often leads to remote calls between application clients/JSPs/servlets and EJBs or between EJBs. Such remote calls are costly and affect the performance of the application as a whole. An increase in the number of such remote calls increases network traffic too. Moreover, for all the advantages that EJBs offer, they come with a small price. The following three design patterns suggest good solutions to minimize some of the performance costs in typical J2EE applications.

Fast Lane Reader

Problem Domain

Most Web applications need to display a list of data objects to the user. A financial service organization that offers its service over the Web will have to display its catalog of services to the customer who is visiting its Web site. An online banking application will have to display a list of recent transactions of a customer who is checking his or her account. If an organization has an internal Web application that allows its employees to enroll for benefits over its intranet, the application should display a list of benefits available to an enquiring employee.

Let's start with the financial organization's scenario. To display the catalog of services, typical designs might have an EJB that represents the whole catalog. There might be a CatalogEJB that implements all required business logic related to the catalog, such as adding a new service, removing an existing service or giving a list of service, details to the requesting components. In the case of other components requesting a list of service details, the requesting component will have to get a reference to the CatalogEJB, access the EJB's remote method that gets the list of services, and then displays (see Listing 1, which assumes the servlet is the client).

This implementation method, while providing transactional access for the required list of objects, comes with the price of the overheads and remote calls associated with the use of the EJB. Given that the list of services is not going to change every second, the chances of the customer seeing stale data are low. So do we have to choose transactional access through the EJB and pay the associated price? Extending the same argument to the other examples, given that the bank customer's list of transactions does not change every second, do we have to have transactional access for listing the transactions? And given that the list of benefits given by the employer does not change every day, do we need transactional access to display the list of benefits to enquiring employees?

Suggested Pattern

The main issue to be addressed here is whether we should choose transactional access while reading a set of data that does not change rapidly. In such scenarios, the suggested pattern would be the Fast Lane Reader. Note that the problem domain that was described requires only read access to a set of data that does not change rapidly. We can opt to take the fast lane and read the data directly rather than opting for transactional access through EJBs. In the scenarios described in the previous section, the Fast Lane Reader will be a more efficient way to access data. Such a technique will also give a faster response to the customer while running a very small risk of displaying stale data. By bypassing the EJB the overheads associated with remote calls, transaction management, and other issues are avoided.

Again taking the financial services scenario and the sample code explained in Listing 1, one way to implement this pattern would be to have a separate object, the Data Access Object, which encapsulates all data read and update functionality of the CatalogEJB (see Listing 2).

A servlet client that uses the Fast Lane Reader pattern will use the Data Access Object directly to read the list of services and display the list as shown in Listing 3.

Since we now have a Data Access Object that encapsulates all access to the catalog, the business methods of the CatalogEJB (like updateServices, addServices, etc.) can use the corresponding methods in Data Access Object to avoid duplication of code.

Comparing Listings 1 and 3, we can see how the servlet bypasses the EJB by accessing the Data Access Object directly to get the list of services from the Catalog. This method of implementation, while allowing the client to use the fast lane for direct access of data, will also allow the EJBs to provide transactional updates to the same data. Note that encapsulating all types of data access will also enable the EJB to use the same Data Access Object for updating the same table. This has the added advantage of avoiding code duplication.

Points to Note

- This pattern is good for faster and more efficient data retrieval only.
- There is a risk of the data being stale and hence this pattern should not be used when the data being accessed changes rapidly or when the tolerance for slightly stale data is very small.
- Because this pattern bypasses the EJB model and provides direct access for persistent data, the design of the application may become complex for large-scale applications.
- As the pattern name suggests, this pattern is ideal for data read. For updating the data, this pattern should not be used as transactional access is bypassed.

Page-by-Page Iterator

Problem Domain

Now that we have seen an efficient way to access read-only data (that does not change rapidly), how much of such data should we read? To put it in a more general way, when we have to access a large remote list of objects, do we transfer the entire list or only part of it? The answer to this question will have a tremendous performance effect on the application because transfer of large lists means extra load on the network. Moreover, if EJBs were used to read such large lists, the performance cost is even more because of the costs introduced by the additional services provided by the EJB model.

To take a concrete example, let us revisit the financial services example that we saw in the previous pattern. Let us assume that the list of services is large. We already saw how we can use the Fast Lane Reader pattern to avoid the overheads associated with EJBs. But, since the list is large, do we transfer the entire list to the requesting client every time? Do we have to stress the network every time a client requests the list of services? If we transfer the entire list every time, do all clients have the capacity to handle such large lists?

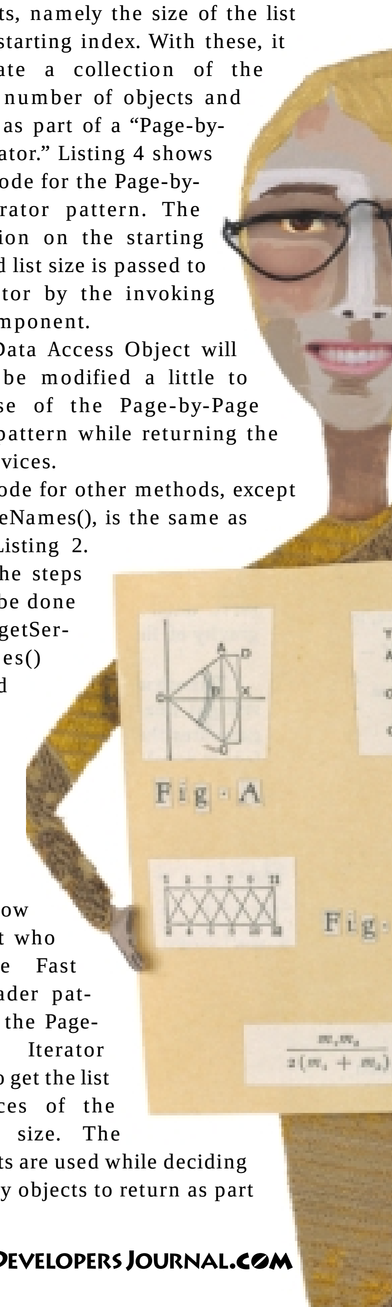
Suggested Pattern

In finding a common and efficient solution, we should note that the user may not be interested in looking at the entire list. The transfer of large lists means more time, resulting in bad user experience. Moreover, the requesting client may have a small amount of memory or a small display; hence, such a client may not be able to handle such large lists. Taking these into account, the suggested pattern would be the Page-by-Page Iterator pattern. The use of this pattern will enable the client to request the list and also specify its size. In response, the server will return a list of the requested size only.

Going back to the example we saw in the previous pattern, implementation of this example will require the getServiceNames() method of Data Access Object (see Listing 2) to take two arguments, namely the size of the list and the starting index. With these, it will create a collection of the required number of objects and return it as part of a "Page-by-Page Iterator." Listing 4 shows sample code for the Page-by-Page Iterator pattern. The information on the starting index and list size is passed to the iterator by the invoking client component.

The Data Access Object will have to be modified a little to make use of the Page-by-Page Iterator pattern while returning the list of services.

The code for other methods, except getServiceNames(), is the same as that in Listing 2. We see the steps that will be done in the getServiceNames() method instead of going into detailed code. These changes will allow the client who uses the Fast Lane Reader pattern and the Page-by-Page Iterator pattern to get the list of services of the required size. The arguments are used while deciding how many objects to return as part



of the collection (see Listing 5). And the servlet client that gets and displays the list would look like Listing 6.

Points to Note

- Clients can specify the list size so they control the amount of data they want to receive. This gives them more flexibility.
- While this pattern provides efficient access to large server-side lists, the number of server hits might increase a bit as not all data is transferred at once.
- The iterator does not keep its own copy of the list being traversed. As a consequence, insertions or removals will interfere with the traversal. But this may not be a problem if the client browses static data collections like list of services or search results.
- Network bandwidth is not wasted by transmitting unused data. This has to be weighed against the potential increase in number of server hits while deciding whether to use this pattern.

Value Object

Problem Domain

So far we have seen two patterns that could improve the efficiency and performance of our J2EE application when large amounts of objects are being transferred between clients and server. All will be well if the actual objects that are being transferred have no attributes. But more often than not, the data being transferred are collections of objects that have their own identity. In the financial services sample application that we have been discussing, the list of services being returned from the server are actually collections of objects. Each of these services will have its own attributes, such as Service Name, Service Cost, Service Description, and Terms and Conditions. And, more often than not, a client requesting the list of services will also want to know about all the above mentioned information that belongs to a service. Given these circumstances, when do we transfer these finer details? It is definitely not efficient for the client to get a list of service names and then contact the server to get the finer details of each service name received. That would entail lots of remote calls and additional stress on the network.

Suggested Pattern

While coming up with a common solution to this scenario, we have to keep in mind that:

- The probability of the client asking for the finer details of a service is much more than the client asking for the complete list of services.
- All the attributes associated with the requested object will most probably be in the same database table and hence can be obtained in the same SQL call that gets the list of service names.

The suggested solution would be that we transfer coarse-grained data (in this example, the list of service names along with all its attributes) rather than transferring fine-grained data (e.g., requiring the client to make separate remote calls for each of the attributes, such as service name, cost, and terms). We can do this by using the Value Object pattern. Use of this pattern in our sample case will result in the ServiceDataAccessObject aggregating each service and all its attributes into one instance of a Value Object, say ServiceInformation. Then a collection of these Value Objects is serialized and sent over the wire to the client where it will be deserialized and used.

To see some code examples, let us first define the ServiceInformation Value Object (see Listing 7).

Now for the rest of the code: the iterator pattern will be the same as that in Listing 4, without any change. The data access object will also be the same as in Listing 5. The Data Access Object will set the iterator's collection as a set of value objects and the servlet will retrieve the attribute information from value objects and use it for display. The code of the servlet client that gets the value objects and displays the values would look like Listing 8.

Points to Note

- Use of this pattern reduces network traffic and improves response time for coarse-grained, read-only data.
- To further improve perfomance, the value objects can be cached.
- Use of this pattern also reduces the hits on server as the number of remote calls made by clients is reduced.
- The extra classes that represent various objects and attributes as Value Objects may add to complexity but this is a small price to pay when we consider the advantages of using this pattern.
- The Value Objects must be serializable and immutable. If clients have the capability to modify the Value Objects, then the state of the objects with the client and that with the server differ, leading to inaccurate states. But the consequence of making the Value Objects immutable is that, if the data changes rapidly, the values in the Value Object might be stale
- Although, in the sample code, we saw the Value Objects being used for transferring information from server to client(s), this pattern can be used for data transfer in the reverse direction also.

oak grove systems

Need a
JAVA Workflow
Solution?

reactor 5

The Process Engine for Enterprise JAVA Applications

Platform & Database Independent

J2EE Application Server Neutral
- BEA WebLogic, IBM WebSphere, JBOSS...

XML/HTML In/Out for Interoperability
- With e-business applications
- With ERP, CRM & legacy systems
- With other workflow systems

Flexibility & Speed
- Faster workflow development
- Quicker process modifications & changes
- Easier reuse of processes
- Simpler integration

DOWNLOAD @
www.oakgrovesystems.com

Download A Fully Functional Reactor 5.0 Evaluation Toolkit at www.oakgrovesystems.com

Contact Information
info@oakgrovesystems.com
626-296-6390 *(phone)*
877-777-6310 *(toll free)*
www.oakgrovesystems.com

Conclusion

In this article we saw three different patterns that we can use effectively to improve the efficiency, performance, and user experience of a J2EE application. The samples we saw to exemplify the use of the patterns seems to combine use of all three. But we must understand that there is no need to use all these three patterns together. For example:

- We can use the Value Object to send a single coarse-grained object (like contact information of a customer) from the server to the client(s).
- We can use the Page-by-Page Iterator to send lists of simple objects from EJBs to clients.
- We can use the Fast Lane Reader to read data from the server and display all of them in one shot.

The sample code we saw was also incomplete. The intention was just to give you an idea of what we are talking about. The Java Pet Store sample application and the patterns catalog of the J2EE Blueprints program provide formal definitions of the patterns we discussed along with full-blown, working sample code, UML diagrams, and other helpful information. Interested readers may find more detailed information on the Web site, http://java.sun.com/j2ee/blueprints. The next version of Enterprise JavaBeans (EJB 2.0) removes some of the overheads associated with remote calls by introducing the concept of "local interfaces." In a future article, we can discuss the effect of such a feature on these patterns.

References

1. *Design Patterns section of the J2EE Blueprints Program*: http://java.sun.com/j2ee/blueprints
2. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns.* John Wiley & Sons Ltd.
3. *M. Fowler's Information System Architecture*: http://martinfowler.com/isa/index.html
4. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.
5. *"Gang of Four" Patterns Template:* http://hillside.net/patterns/Writing/GOFtempl.html

AUTHOR BIO

Vijay Ramachandran, a member of the J2EE Blueprints team for Sun Microsystems, contributed to the recent Java Pet Store sample application. He's currently focusing on the Web services features of forthcoming J2EE releases.

vijay.ramachandran@sun.com

Listing 1: Servlet that uses a CatalogEJB to read a list of objects

```
// All required imports

public class DisplayServices extends HttpServlet {

    // variable declarations

    public void init() {
        // all inits required
    }

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

        // other processing, if any

        InitialContext ctxt = new InitialContext();
        Object obj = ctxt.lookup("java:comp/env/ejb/Catalog/Services");
        CatalogHome catHome = (CatalogHome)
                    PortableRemoteObject.narrow(obj, CatalogHome.class);
        Catalog catList = catHome.create();
        Collection  list = catList.getServiceNames();
        while(int i=0; i < list.size(); i++) {

            // get the service name

            // display the service name received

        }
        // do other processing required
    }
    // other methods definitions
}

// The following is the sample code of the CatalogEJB
// For simplicity sake, the definitions of the home and remote
// interfaces are not shown

// All required imports

public class CatalogEJB implements EntityBean {

    // Methods like ejbCreate etc go here

    // Business methods start here
    public Collection getServiceNames() {
        // Access the database and get the list of service names
        // Form a collection of the service names
        // Return this collection
    }

    public int updateServices(arg1, arg2) {
        // this method will allow the CatalogEJB to update the list of
             Services;
        // for example, this method might build the SQL query for updating
             the
        // list services in the catalog and use JDBC to execute the query
    }

    public int addServices(arg1, arg2) {
        // this method will allow the CatalogEJB to add to the list of
             Services
    }

    // Other business methods
}
```

Listing 2: Data Access Object that encapsulates all types of access to the Catalog

```
// all required imports

public class CatalogDataAccessObject {

    private Connection dbConnection;
    private DataSource datasource;

    public CatalogDataAccessObject() {

        // appropriate exceptions to be caught
        InitialContext ic = new InitialContext();
        datasource = (DataSource)
                ic.lookup("java:comp/env/jdbc/CatalogDataSource");
        dbConnection = datasource.getConnection();
    }

    public int updateServices(arg1, arg2) {
        // this method will allow the CatalogEJB to update the list of
             Services;
        // for example, this method might build the SQL query for updating
             the
        // list services in the catalog and use JDBC to execute the query
    }

    public int addServices(arg1, arg2) {
        // this method will allow the CatalogEJB to add to the list of
             Services
    }

    public Collection getServiceNames() {
        // this method will allow the catalogEJB or any client
             using Fast Lane
        // Reader pattern to get the list of services
    }

    // other method definitions as required
}
```

Listing 3: Servlet that uses Fast Lane Reader pattern

```
// All required imports

public class DisplayServices extends HttpServlet {

    private CatalogDataAccessObject dao;
    // other declarations

    public void init() {
        CatalogDataAccessObject dao = new
          CatalogDataAccessObject();

        // all other inits required
    }

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

        // other processing, if any

        Collection list = dao.getServiceNames();
        for(int i=0; i<list.size(); list++) {

            // get the service name received
            // display the list name
        }

        // do other processing as required
    }

    // other method definitions, if any
}
```

Listing 4: Page-by-Page Iterator pattern returned to the client

```
// all required imports

public class PageByPageIteratorImpl {

/*
"start" represents the starting index of the first element of this
   sublist.
*/
    private int start;

/*
"size" represents the number of elements that are in this sublist.
*/
    private int size;

/*
"totalCount" represents the total number of objects that would
   have been
returned in the absence of the Page-By-Page Iterator; very useful
   information
for the clients in deciding to iterate more or not
*/
    private int totalCount;

/*
"objs" has the actual collection of objects
*/
    private Collection objs;

    public PageByPageIteratorImpl(int start, int size, int total,
      Collection
coll) {
        this.start = start;
        this.size = size;
        this.totalCount = total;
        this.objs = coll;
    }

    public int getStartIndex() {
        return start;
    }

    // Other similar accessor methods for size, totalCount, objs
}
```

Listing 5: Data Access Object

```
    public PageByPageIteratorImpl getServiceNames(int startIndex,
      int size) {
        // get the total count of the number of service into
      variable "total";
        // access the database and get the collection of service
             names in
        // "retObjs";
        // the collection obtained will be of requested size only
             starting from
        // the index specified;
        // return the collection of service names as part of the
             Page-by-Page Iterator
        return(new PageByPageInteratorImpl(startIndex, size,
             total, retObjs));
    }
    // other method definitions as required
}
```

Listing 6: Servlet using Fast Lane Reader and Page-by-Page Iterator

```
// All required imports

public class DisplayServices extends HttpServlet {

    private CatalogDataAccessObject dao;
    // other declarations

    public void init() {
        CatalogDataAccessObject dao = new
          CatalogDataAccessObject();
        // all other inits required
    }

    public void doPost(HttpServletRequest request,
      HttpServletResponse response)
            throws IOException, ServletException {
        // other processing, if any
        Integer startIndex = new
          Integer(request.getParameter("StartIndex"));
        Integer listSize = new
          Integer(request.getParameter("ListSize"));
        PageByPageIteratorImpl list =
          dao.getServiceNames(startIndex.intValue(),
                                            listSize.
                                              intValue());
        Collection objs = list.getCollection();
        // Now the list received is of requested size only
        // iterate thro the list and display the service names
             received
        // do other processing as required
    }
    // other method definitions
}
```

Listing 7: Value Object class for a service

```
// all imports required

public class ServiceInformation implements java.io.Serializable {

    private String serviceName;
    private double serviceCost;
    private String[] serviceTerms;
    // other attributes of a service

    public ServiceInformation(String name, double cost, String[]
      terms) {
        serviceName = name;
        serviceCost = cost;
        serviceTerms = terms;
    }

    public String getName() {
        return(serviceName);
    }

    public double getCost() {
         return(serviceCost);
    }

    public String[] getTerms() {
         return(serviceTerms);
    }
    // similar accessor methods for other attributes
}
```

Listing 8: Using all these patterns discussed

```
// All required imports

public class DisplayServices extends HttpServlet {

    private CatalogDataAccessObject dao;
    // other declarations

    public void init() {
        CatalogDataAccessObject dao = new
          CatalogDataAccessObject();
        // all other inits required
    }

    public void doPost(HttpServletRequest request,
      HttpServletResponse response)
            throws IOException, ServletException {
        // other processing, if any
        Integer startIndex = new
          Integer(request.getParameter("StartIndex"));
        Integer listSize = new
          Integer(request.getParameter("ListSize"));
        PageByPageIteratorImpl list =
          dao.getServiceNames(startIndex.intValue(),
                                        listSize.intValue());
        Collection objs = list.getCollection();
        // The collection is a collection of the value objects
        // Now the list received is of requested size only
        // extract the service names and the attributes from the
             value objects
        // display everything and do other processing as required
    }
    // other method definitions
}
```

JEREMY GEELAN J2SE EDITOR

The New Cycle Arrives

A new business cycle is sweeping the Internet technology world, one that now demands that companies start competing with each other, not only for new customers, but also – and perhaps even more crucially in a time of shrinking revenues – to retain the ones they have.

In this new cycle Java as usual acts as a focal point, because in such a phase the overriding need is for an enterprise to contradistinguish itself from its competitors, and what better way to do so than by providing not just better customer support for existing products or services (everyone tries that), but also entirely new services.

This is, of course, more or less where Java came into the Internet technology arena in the first place. Developing a customer-facing, product-support strategy is one thing, but actually delivering a great customer experience is another, and some of the articles in this month's J2SE section directly address the bumps in the road that lead to successful Java-based implementations.

José María Barrera looks at the best way to use the Java reflection classes; Mark Dykstra examines how to eliminate multithreaded errors; and Thomas Hammell invites us to "drag-and-drop" into Java. These writer-developers, and others like them, are what make ***JDJ*** the leading print and online resource for Java developers in the new business cycle as in the old.

Getting the Edge on Web Services

One of the liveliest new e-business battlegrounds will undoubtedly be Web services. Those of you reading this issue of ***JDJ*** prior to the **JDJEdge 2001 Conference & Expo**, which is being presented in New York later this month by **SYS-CON Events,** may already have made up your mind to sample some of the sessions at the **Web Services Edge Conference & Expo** that SYS-CON is colocating with **JDJEdge** at the New York Hilton.

If so, you'll have the benefit of hearing firsthand the views of those in the forefront of the paradigm that promises to do for distributed computing what Captain Cook did for Botany Bay: put it on the map. If not, try and come even if it's only at the last minute: there isn't an event anywhere like it on the East Coast. Web services will be the theme of a keynote discussion panel that the two conferences will be holding.

So if you're wondering what James Gosling thinks about it all, or what a Java-based vendor such as PointBase is doing in this "brave new Web services world", or where BEA Systems' Scott Dietzen thinks it's all headed, try and make it to the N.Y. Hilton, September 23–26.

While there, you'll also have an opportunity to savor keynote addresses from such luminaries as David Litwack, CEO of SilverStream; Gregg Kiessling, cofounder and CEO of Sitraka Software; Yogesh Gupta, CTO of Computer Associates; and Dr. Alan E. Baratz, CEO of Zaplet, Inc. – which, at an event already offering Kevin Lynch, president of products, Macromedia Inc., and Rick Ross, president of the 54,000-strong Java Lobby, is quite a lineup.

If you haven't had the pleasure of hearing Sun's James Gosling speak, remember that he's not only the "father of Java," but has also built satellite data-acquisition systems, a multiprocessor version of UNIX, several compilers, mail systems, window managers, a WYSIWYG text editor, a constraint-based drawing editor – and, oh, yes, the Emacs text editor for UNIX systems. When it's time for audience questions, be sure to ask him just what lies inside the cover of his PhD thesis, "The Algebraic Manipulation of Constraints."

For us mere mortals who never made it to Stanford or Carnegie Mellon in the crucial late '70s when Internet technology was in its incubation, **JDJEdge 2001** promises to be a great eye-opener! These are truly the Java movers and shakers, and I for one wouldn't miss it for the world.

jeremy@sys-con.com

AUTHOR BIO

Jeremy Geelan, editorial director of SYS-CON Media, speaks, writes, and broadcasts frequently around the world about the future of Internet technology and about the business strategies appropriate to the convergence of business, i-technology, and the future.

J2SE INDEX

Forte for Java 3.0

by Sun Microsystems

REVIEWED BY JIM MILBERY
jmilbery@kuromaku.com

specs

Sun Microsystems Inc.
901 San Antonio Road
Palo Alto, CA 94303
Web: www.sun.com/forte/ffj

Test Environment
Dell 410 Precision CPU (450MHz), Windows NT Workstation Service Pack 5, 256MB RAM

Specifications
Platforms: Solaris 8
Red Hat Linux 6.2
Windows 98 NT/2000
Pricing: Community Edition freely downloadable. Enterprise Edition is $1,995/seat, 30-day trial is freely downloadable.

In my opinion there have always been two types of Java application developers. The first type prefers to use a text editor, compiler, and debugger to get the job done. Once upon a time, this was the only way to write code, from COBOL and Fortran all the way through C. The age of the fourth-generation language introduced the concept of a specialized developer "coding tool," which we now refer to as an integrated development environment (IDE).

The second type of programmer prefers to use an IDE to develop Java code. There's no right or wrong way here – it's all a matter of preference. Initially, most of the Java IDEs on the market supported the Windows operating system as a development platform, but newer Java IDEs are written in Java. This makes them highly portable from platform to platform. Sun offers its own such IDE in the form of Forte for Java. I recently looked at the Early Access Edition of Forte for Java, 3.0 release, Enterprise Edition.

Forte for Java — Two Editions

Sun's Forte for Java is based on two sets of technology that Sun acquired, Forte Software's Forte and NetBeans' NetBeans Developer. Forte Software predates Java. They were one of the first companies to tackle the complex issue of distributed application processing. NetBeans was a Czech-based company that developed one of the first IDEs written entirely in Java. Sun acquired Forte Software for Forte's server-side technology and code-generation expertise and they bought out NetBeans for their Java IDE. Forte for Java – or FFJ – is the resultant offspring of these two exciting technologies.

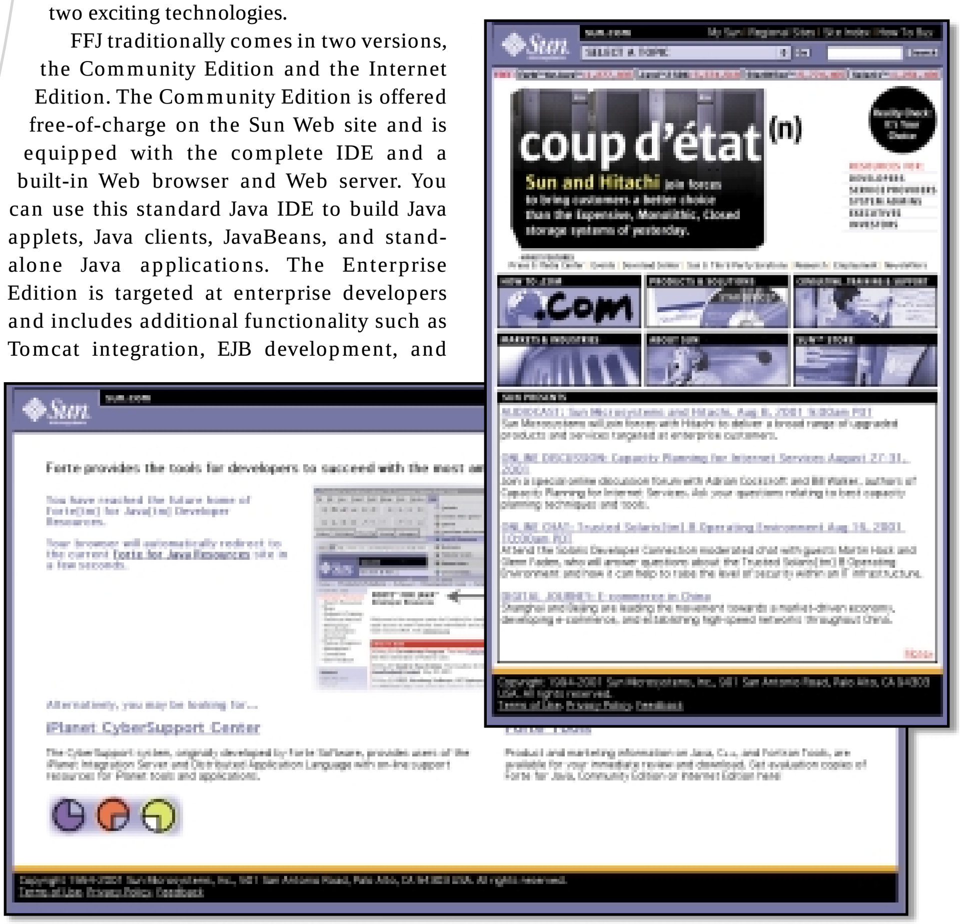
FFJ traditionally comes in two versions, the Community Edition and the Internet Edition. The Community Edition is offered free-of-charge on the Sun Web site and is equipped with the complete IDE and a built-in Web browser and Web server. You can use this standard Java IDE to build Java applets, Java clients, JavaBeans, and standalone Java applications. The Enterprise Edition is targeted at enterprise developers and includes additional functionality such as Tomcat integration, EJB development, and database-aware components. The updated release includes both the Community and Internet Edition functionalities as well as new functionality. FFJ Enterprise Edition comes equipped with modules for EJB creation, application server integration, and an enterprise services presentation toolkit.

Working with FFJ

The Forte for Java Enterprise Edition 3.0 release comes packaged in an installation wizard. The install process is relatively simple and I was able to get the product up and running in a few minutes. FFJ 3.0 has a plethora of features, and it will take you awhile to get a handle on all the ins-and-outs of this comprehensive tool set.

If you are coming to FFJ from another graphical IDE environment, I suspect you'll find most of the panels and forms to be well organized. However, if you're coming from the editor/compiler environment, be prepared to be overwhelmed by the sheer number of choices in the FFJ IDE. This is typical for most Java IDEs. The theory is that the IDE is not really useful unless it covers all possible bases. Java continues to grow in scope and complexity, and experienced Java developers won't be surprised by the complexity of the IDE environment. However, the number of switches and knobs within the IDE might be overwhelming for newly minted Java programmers.

The FFJ 3.0 release comes equipped with a short tutorial and some limited sample code.


sun.com
coup d'état (n)
Sun and Hitachi join forces to bring customers a better choice than the Expensive, Monolithic, Closed storage systems of yesterday.
.Com
SUN PRESENTS
iPlanet CyberSupport Center

AUDIOCAST: Sun Microsystems and Hitachi, Aug 8, 2001 9:30am PDT
Sun Microsystems will join forces with Hitachi to deliver a broad range of upgraded products and services targeted at enterprise customers.

ONLINE DISCUSSION: Capacity Planning for Internet Services August 27-31, 2001
Join a special online discussion forum with Adrian Cockcroft and Bill Walker, authors of Capacity Planning for Internet Services. Ask your questions relating to best capacity planning techniques and tools.

ONLINE CHAT: Trusted Solaris[tm] 8 Operating Environment Aug 15, 2001 10:00am PDT
Attend the Solaris Developer Connection moderated chat with guests Martin Hack and Glenn Faden, who will answer questions about the Trusted Solaris[tm] 8 Operating Environment and how it can help to raise the level of security within an IT infrastructure.

DIGITAL JOURNEY: E-commerce in China
Shanghai and Beijing are leading the movement towards a market-driven economy, developing e-commerce, and establishing high-speed networks throughout China.

Copyright 1994-2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA. All rights reserved.
Terms of Use. Privacy Policy. Feedback

Forte provides the tools for developers to succeed with the most

You have reached the future home of Forte[tm] for Java[tm] Developer Resources.

Your browser will automatically redirect to the current Forte for Java Resources site in a few seconds.

Alternatively, you may be looking for...

iPlanet CyberSupport Center
The CyberSupport system, originally developed by Forte Software, provides users of the iPlanet Integration Server and Distributed Application Language with on-line support resources for iPlanet tools and applications.

Forte Tools
Product and marketing information on Java, C++, and Fortran Tools, are available for your immediate review and download. Get evaluation copies of Forte for Java, Community Edition or Internet Edition here!

Copyright 1994-2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA. All rights reserved.
Terms of Use. Privacy Policy. Feedback

Forte for Java 3.0 by Sun Microsystems

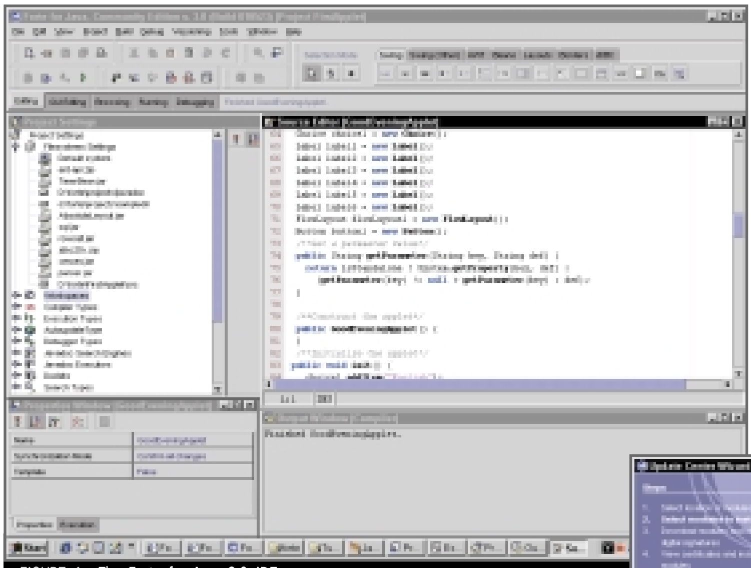


FIGURE 1 The Forte for Java 3.0 IDE

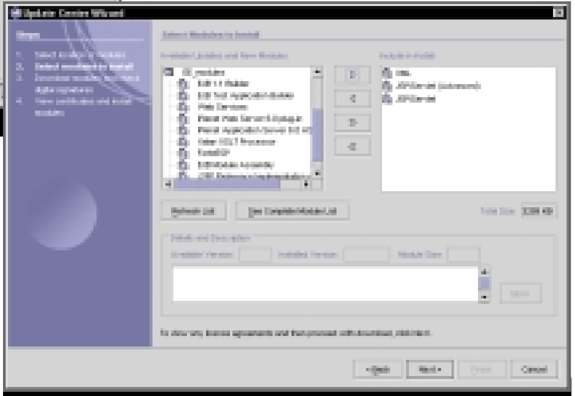


FIGURE 2 FFJ Update Center

If the sample projects don't appeal to you, you can use FFJ with your existing code. FFJ 3.0 can input projects from other Java IDEs such as Microsoft J++, JBuilder, and VisualCafé (WebGain). Thus I was able to grab an old JBuilder project I had lying around and move it into FFJ without much difficulty. It's definitely a programmer's IDE. Developers not familiar with Java are likely to get lost in an IDE that's as comprehensive as FFJ. The product includes numerous wizards for managing projects and generating code, but you'd better be familiar with the basics of Java before you dive into creating your own projects. (This is definitely a personal opinion and does not reflect on the capabilities of FFJ.) The question for new Java developers is whether a comprehensive IDE helps or hurts their learning curve. Once you're familiar with Java (even to a limited degree) the value of a powerful IDE is not open to questions.

Forte's wizards include:

- JSP and servlet
- XML and DTD
- AWT forms
- Ant projects
- Beans
- CORBA
- Classes (applets, classes, etc.)
- Database (forms and database schema creation)
- Sample forms
- JAR packaging
- RMI
- Swing forms

FFJ uses the familiar "workspaces" and project concepts, so it's a simple process to organize your tasks and work on multiple, independent projects at the same time. You can organize the various task panels (as shown in Figure 1) to match your own preferences, and dock and undock windows as well. Through the Form Editor you can design the graphical display and view/set properties in the Component Inspector, and modify the resulting Java code in the Source Editor.

Although the FFJ environment is somewhat sluggish from a GUI perspective, the source code editor has some nice features, including syntax highlighting and dynamic code completion. You can change the fonts and colors to match your own personal preferences. (Opinions vary on the utility of code completion, but I make use of this feature extensively when writing code.) Within the GUI editing environment Forte provides a "connection capability" for generating code. For example, you can add a button to control a chart object by dropping the button on the form and using the connection wizard to link the button to the chart. Forte's Connection Wizard walks you through the process of selecting events and generating the proper Java code. Sun provides a valuable demonstration of these two features (and more) on the Forte Web site, and I would encourage you to take this short online tour.

When it's time to test your code, the debugging workspace comes into play. It's nicely laid out and provides all the standard Java debugger capabilities (including remote debugging and JPDA) – and you can customize the debugger for your environment as needed.

One of the nicest features that I worked with is the Update center (see Figure 2). It's available to all registered Forte for Java users and provides a library of updates and modules that can be plugged directly into the FFJ IDE. These modules can include everything from EJB builders to application server interfaces. This is exactly the type of feature that leverages the power of the Internet. Developers need not stuff their local machine with a whole host of add-ons and features they don't need. Rather, they can connect to a central site and download functions as necessary – making the Forte IDE highly extensible and customizable.

JDJ Product Snapshot

- ***Target audience:*** Java Developers
- ***Level:*** Mid-level to advanced
- ***Pros:*** Multiplatform support for IDE, extensibility of the IDE and Web-based updates, integration with Sun ONE
- ***Cons:*** IDE tends to be sluggish, Java newbies may be overwhelmed with all of the features

Summary

FFJ 3.0 is a strong offering in the Java IDE category. Its ability to run on multiple operating platforms and support Internet-based updates will find favor with most Java developers. Java newbies may find the IDE slightly overwhelming, but experienced Java developers will appreciate the wealth of features and functions within FFJ.

A Reusable Drag and Drop Handler

Simplifying the implementation of the drag and drop

WRITTEN BY
THOMAS HAMMELL

The ability to transfer information by dragging data from one component to another has been around since the development of the graphical user interface. Over the years drag-and-drop has gone from a cool feature to a required piece of most user interfaces. Most users expect to be able to drag objects between fields, windows, or folders and have some action occur. Drag-and-drop is even used to open applications by dragging a file to an application icon.

For a Java developer creating user interfaces it's no longer a question of *whether* drag-and-drop should be used but of *how much*. Java provides a set of classes for implementing the drag-and-drop interface. While it's not overly complex, implementing it in a complex GUI with a number of drag sources and drop targets can be tedious and error-prone. In this article, I develop an abstract DnDHandler class that takes care of most of the tedium and simplifies the implementation of drag-and-drop.

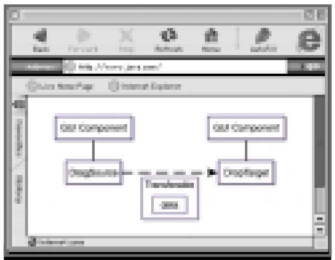


FIGURE 1 Drag-and-drop transaction

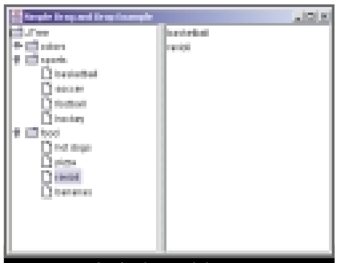


FIGURE 2 Simple drag-and-drop app

Drag-and-Drop Review

Although there are many parts to a drag-and-drop transaction, it can basically be broken down into three main components (see Figure 1):

1. Starting the drag where the drag action is recognized by the component
2. Converting the drag item into a transferable data type
3. Dropping the transferable data into the drop target

During the drag-and-drop transaction many other minor parts allow for user feedback, but these won't be discussed in any detail in this article. For more information on the details of drag-and-drop, see the reference at the end of the article.

Consider the simple drag-and-drop application shown in Figure 2 that contains two components: a DraggableTree on the left and a DroppableList on the right. The user can select items from the tree and drag them into the list.

The code for the DraggableTree is shown in Listing 1 and the code for the DroppableList is shown in Listing 2. Listings 1–5 can be downloaded from www.sys-con.com/java/sourcec.cfm.

To make the tree a draggable component a number of things must be done. First, the tree must implement the DragGestureListener interface, then create an instance of a DragSource and call the createDefaultDragGestureRecognizer method so that the tree will be notified when a drag action has been initiated.

When a drag-and-drop action occurs, the dragGestureRecognized method is called. This method first checks to see if something has been selected. If it has, the method then gets the selected object, creates a transferable version of the data, and calls the start drag function.

The list that acts as the drop target must implement the DropTargetListener interface and create a DropTarget instance. The DropTarget constructor is used to notify the drag-and-drop framework that the list will accept dragged objects.

Although a number of methods are defined in the DropTargetListener interface, the main one is the drop method, which is called when an object is dropped into the list.

The drop method gets the transferable data from the DropTargetDropEvent. If the transferable data is the right data type, a number of steps are taken. First, the drop is accepted. Next, the transferred data is retrieved and the item is added to the list. Last, the dropComplete function is called to notify the drag-and-drop framework that the drop was completed successfully.

There's a lot more to drag-and-drop than presented in this simple example, but the example shows the basic steps in any drag-and-drop transaction.

The DNDHandler Class

Unlike the previous example, implementing drag-and-drop in a more complex GUI or a multiple document interface involves a lot of code duplication, if there are more than a couple of drag-and-drop targets. It's better if all the common code for a drag-and-drop transaction is contained in a single class where it's reused by any GUI component that needs to implement drag-and-drop.

This common drag-and-drop class is called *DnDHandler*. It should contain as much of the drag-and-drop functionality as possible and implement both the drag and the drop interfaces.

The requirements needed to create a draggable GUI component are:

- Implement the DragGestureListener interface
- Create an instance of a DragSource
- Call the createDefaultDragGestureRecognizer method so the component will be notified when a drag action has been initiated
- Create a dragGestureRecognized method that will get the draggable data and start the drag

The only problem with making these requirements into a generic class is that the dragGestureRecognized method must know how to get data from the actual component. To get around this problem, the dragGestureRecognized in the DnDHandler class is changed so it calls a getTransferable method to get the data from the actual component. The DnDHandler class makes this an abstract method so it will need to be implemented by the class extending it.

Now let's look at the requirements for a droppable GUI component:

- Implement the DropTargetListener interface
- Create an instance of a DropTarget instance
- Create a drop method to add the dragged data to the component

Again, the only problem with making these requirements generic is the drop method, which needs to know how to add the data to the dropped component. For the DnDHandler class we'll modify the drop method to call an abstract handleDrop method that needs to be implemented by the class extending the DnDHandler class. The resulting DnDHandler class is shown in Listing 3.

The DnDHandler class implements the DropTargetListener, DragSourceListener, and DragGestureListener interfaces. Its constructor takes the components as an argument and creates an instance of a DragSource and DropTarget and contains three abstract methods:

1. ***getTransferable:*** Gets transferable data from the component
2. ***handleDrop:*** Handles the drop action
3. ***getSupportedDataFlavors:*** Gets the transferable data that's supported

This DnDHandler class encapsulates all the requirements needed to create a drag-and-drop interface.

Using the DnDHandler Class

Let's now rework our original drag-and-drop example using the DnDHandler class and see how this simplifies the implementation. To use the DnDHandler class, the new class that's used to replace DraggableTree needs to extend both JTree and the DnDHandler classes, but Java's single inheritance limitation makes this impossible. To get around this limitation we use an inner class to extend the DnDHandler class, while the main class extends JTree.

The reworked DraggableTree class, DNDTree, is shown in Listing 4. The inner class DNDTreeHandler does most of the work. The only thing the outer class does is create an instance of the DNDTreeHandler. The DNDTreeHandler class has the one argument constructor needed by its parent and the implementation of the three abstract methods.

By implementing the handleDrop, the DNDTree class has some improved functionality over the original version. The tree is now a drop target and items from the list can be dragged onto the tree. This is one of the advantages of using the DNDHandler class: instead of implementing all the methods of a DropTargetListener, we simply write the handleDrop function. If we didn't want the DNDTree to be a drop target, we would still have to implement the handleDrop function, but we could just ignore the drop event or call the rejectDrop method.

The reworked Droppable list class is called DNDList (see Listing 5). It's coded in a similar manner to the DNDTree class.

The reworked example is still a simple application of drag-and-drop but it does show how the DNDHandler calls simplify drag-and-drop implementation. Instead of implementing three interfaces and numerous methods, drag-and-drop can now be executed by writing three simple methods.

Conclusion

The DnDHandler class has proven very useful in our development of user interfaces. It allows us to add drag-and-drop features to the user interface more quickly and also provides a central place to control the low-level workings of drag-and-drop. We're using it under Java 1.3 on both Windows and UNIX and haven't experienced any problems.

Most real-world projects will have many more transferable data types than the simple example presented. The management of these transferable data types is an important implementation issue that must be considered when using drag-and-drop in a complex user interface.

Hopefully this article has taken some of the mystery out of implementing drag-and-drop. Although it looks difficult, it's fairly straightforward using a class such as DnDHandler.

Author Bio

Thomas Hammell is a senior developer at Hewlett-Packard Bluestone and is part of the tools groups that develops various tools for HP middleware products. He has over 15 years of experience in developing software. Tom holds a BS in electrical engineering and an MS in computer science from Stevens Institute of Technology.

"It's no longer a question of *whether* drag-and-drop should be used but of *how much*"

Reference

1. Zukowski, J. (1999). *John Zukowski's Definitive Guide to Swing for Java 2.* Apress.

thomas_hammell@hp.com

Jtest 4.0 by ParaSoft

REVIEWED BY JIM MILBERY
jmilbery@kuromaku.com

specs

ParaSoft
2031 S. Myrtle Ave.
Monrovia, CA 91016
888 305-0041
www.parasoft.com

Test Environment:
Toshiba Satellite Pro 4600, 866 MHz Intel Pentium III processor, 20GB disk, 256MB RAM, Windows 2000 w/Service Pack 2

Specifications:
Platforms: Windows NT/2000/98/ME, Linux, Solaris
Pricing: $3,495/developer

New-car buyers often fear that they're getting a "Friday afternoon" vehicle – a car built by the last shift at the end of a tough week. Manufacturers have spent an untold number of man-years trying to prevent such defects.

As developers you face the same problem: no matter how carefully you work, you'll inevitably make mistakes. The quality assurance department within your organization should be able to catch your mistakes before they make it into production, but they're the last line of defense. Ideally, you'll want to test your code before it makes its way over to the QA department. The sooner you find the problem, the easier it is to fix.

A number of products can help you test your code – but many of these solutions require you to write complicated test routines and scripts. These scripts often get out of sync with the code as project deadlines grow short. ParaSoft offers a fresh alternative with their Jtest 4.0 product.

Jtest Particulars

Jtest, itself a Java application, provides three different types of testing for your Java code:

- Static testing
- White-box testing
- Black-box testing

The simplest of these is static testing. Jtest compares your Java source code against a set of predefined coding rules, checking source code against a variety of rules from over 20 different categories. As a programmer I generally dislike this type of automated source-code checking because it rarely takes into account my personal coding style. However, static testing is an invaluable resource for project managers as it helps ensure that coding styles are consistent across developers and programs. It's much easier to support and maintain code that's been built in a consistent fashion across modules. In this era of "guerilla" software development, such consistency checking is doubly important. Jtest allows you to edit the static testing rules, making the entire process much more palatable. Figure 1 shows the Jtest rules list in the overlay window.

You can pick and choose from the list of prebuilt rules, and you can use the RuleWizard to add customized rules. The complete set is then stored as a file that can be shared among developers. Thus the project manager can configure a single set of rules that can be applied across an entire team of developers. I was able to use Jtest on some sample Java code (as shown in Figure 1) and it quickly pointed out some of the most common mistakes. (Jtest can't access JAR-based code, for example, and you have to provide Java source files for static analysis.)

The second layer of testing offered by Jtest, white-box testing, checks to ensure that a class is structurally sound. (It doesn't attempt to verify that a class behaves according to specification.) With Jtest you may need to create some "test classes" to thoroughly exercise certain types of code (such as EJBs, RMI, and application server integration).

Black-box testing checks that a class behaves according to specification (test cases the user generates). It checks that the class produces the correct output (outcomes) for a given input (or set of inputs). Jtest can derive inputs from your code by using Design by Contract (DbC) instrumentation, and you can also create explicit test classes that Jtest will use to exercise your code. (Jtest's static tests can help you to verify that the DbC assertions are contained within your Java code.)

ParaSoft provides extensive documentation on its Web site, including tutorials, FAQs, and user manuals. I was relatively impressed with the amount of information available there for developers, and I encourage you to take a look. When the time comes to run a trial of the software, you can download it from the site, though you'll need to request a license key for Jtest. The key is machine-specific, that is, you can't use the software on multiple machines without requesting multiple license keys.

I was able to get the software installed quickly and easily. Jtest uses a Java-based user interface, and it's relatively speedy. The UI maps errors and deficiencies back to your Java source code, but it doesn't automatically coordinate the outline controls in the display windows. (You can select an error condition in the results display and Jtest will open up the appropriate detailed results window, but it's up to you to click through the detailed results window to get to the specific line items.) Jtest automatically handles regression testing, so

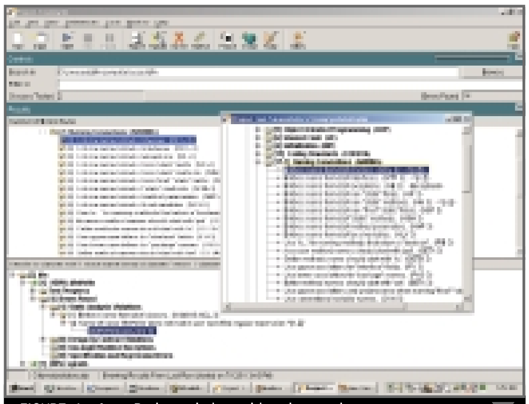


FIGURE 1 Jtest Project window with rules overlay

I was able to make iterative changes to my code and fix errors as I went along.

JDJ Product Snapshot

- ***Target audience:*** Java project managers, Q/A team, Java programmers
- ***Level:*** Entry level to advanced
- ***Pros:*** Comprehensive testing, multiplatform support, reasonably priced
- ***Cons:*** No automated GUI testing, limited integration with popular Java IDEs

Summary

Jtest 4.0 offers a nice platform for building consistent, well-tested Java applications. The integration of static, white-box, and black-box testing makes Jtest an interesting product, but to harness the complete power of Jtest you'll need to invest in the Design by Contract programming methodology.

WHAT IS JAVA

REFLECTION

The spirit and technical side of this advanced Java feature

Written by José María Barrera

welcome to the Java Reflection universe. Once you've been there, you'll never think about programming the way you used to.

Imagine that you're a C++ programmer and you have to implement the following program:

1. Ask for a class name.
2. Create an object of that class.
3. Show the field names for the object and their values.

Ouch! As a C++ programmer you'll start thinking about all the tables or if-then-else-if lists you'll have to create. In Java? Forty lines of code, tops! How come? Reflection uses runtime support not present in languages like C or C++. The kinds of things you can do with Reflection can't be done in those languages.

Reflection is a way of thinking; it's a metalanguage that enables you to analyze and manipulate your objects in a dynamic way. Once you see its possibilities, the sky's the limit: serialization, expression evaluation, language interpretation, class factories, object description, plug-in architectures – you name it. Reflection is one of the most exciting features of Java.

Big industrial-strength protocols like SOAP and JavaBeans wouldn't be possible if it weren't for Reflection. Every time you drag-and-drop an object in your favorite IDE into a form, Reflection is orchestrating the action behind the scenes. Actually, most sophisticated Java applications rely on Reflection in one way or another.

Reflection is an advanced feature of the Java environment. It gives runtime information about objects, classes, and interfaces. Reflection answers questions like:

- Which class does an object belong to?
- What is the description of a given class name?
- What are the fields in a given class?
- What is the type of a field?
- What are the methods in a class?
- What are the parameters of a method?
- What are the constructors of a given class?

Reflection also lets you operate on objects and do things like:

- Constructing an object using a given constructor
- Invoking an object's method using such-and-such parameters
- Assigning a value to an object's field
- Dynamically creating and manipulating arrays

Now you know the spirit behind Reflection. Let's explore its technical side.

Java Reflection Classes

With the exception of the class Class that resides in the default Java package, all Reflection classes are contained in the package java.lang.reflect.

Classes are represented by the class Class, class Fields by the Field class, methods by the Method class, constructors by the Constructor class, and arrays – you guessed it – by the Array class.

Class

Every class and interface in Java is described by a Class object. There are methods in Class to get all the information about the class: name, parent class, constructors, fields, methods, interfaces implemented, and so on.

To obtain the class that an object belongs to, you call the method Class getClass(). This method is defined in the Object class (root of the Java classes hierarchy) and is therefore available to any object.

```
String myString = "my string";
Class theClass = myString.getClass();
```

Every class in Java has a property ".class" that returns a Class object for the class.

```
if (myString.getClass()==String.class)
 System.out.println("The object is a String");
```

Primitive types such as int or Boolean are represented by Class objects as well. The wrapper classes (Integer, Boolean, Double,...) contain a ".TYPE" property that returns the Class object representing the primitive type. Class Class highlights are shown in Table 1.

```
Class myClass = Integer.TYPE;
```

Field

The Field class describes the different attributes of a Java class field. From a Field object you can get the field name, its type, and its accessibility. It also contains methods to set and get the field's value for a given object (see Listing 1). Class Field highlights are given in Table 2.

Method

The Method class allows you to get information about class methods. You can get the method name, its type, its accessibility, and its parameter types. You can also invoke the method on a particular object and pass a set of parameters to it (see Listing 2). Class Method highlights are given in Table 3.

Constructor

The Constructor class allows you to get information about class constructors such as parameter types, number of parameters, and accessibility. It also lets you invoke the constructor to create new object instances (see Listing 3). Class Constructor highlights are shown in Table 4.

Disadvantages and Misuses

I agree that it isn't straightforward to think about thinking. Using Reflection isn't easy at the beginning. The model is simple, but you're using objects called *Object*, classes called *Class*, methods called *Method*.... It takes time to get used to it, but believe me, once you get comfortable with the model, what you can do with Reflection is amazing.

• • •

Up to now I've deliberately avoided the subject of exception handling. Almost every Reflection method throws exceptions, making the code very confusing. Not helping the situation is the wrapping/unwrapping of primitive types. What I've done to alleviate this is to create a ReflectionUtilities library that hides all the implementation details and lets me concentrate on my reflective task.

The methods I've introduced so far to access class members work only on public members, by default. If this weren't the case, you could fool the VM and access members illegally, jeopardizing the security of the system. You can change the default behavior, but that implies that you have the right to do so, which isn't usually the case on Web-delivered applications. This forces you to have to declare the class members that you want to expose to Reflection as public. Object-oriented advocates will tell you that this can violate the encapsulation principle.

Class forName(String className)	Returns a Class object for a given class or interface name. The name must be fully qualified.
Field[] getFields()	Returns all public-accessible fields of the class or interface.
Field getField(String fieldName)	Returns a public Field object for a particular fieldName in the class or interface.
Method[] getMethods()	Returns all public-accessible methods of the class or interface.
Method getMethod (String methodName, Class[] parameterTypes)	Returns a Method object given its name and parameter types. Java accepts method overloading, which means you can have multiple methods with the same name and different formal parameters.
Constructor[] getConstructors()	Returns all public-accessible constructors of the class.
Constructor getConstructor (Class[] parameterTypes)	Returns a Constructor object representing the class constructor that has the given list of formal parameter types.
boolean isAssignableFrom (Class classParameter)	Can the class or interface represented by classParameter be converted into the class or interface?

TABLE 1 Class Class highlights

Class getType()	Returns a Class object representing the field type.
Class getDeclaringClass()	Returns the Class object that represents the class or interface that declared the field.
String getName()	Returns the field's name.
Object get(Object object)	Returns the field's value of a given object. For static fields you can pass null as the object.
void set(Object object, Object value)	Sets the field's value of a given object. For static fields you can pass null as the object.
int getInt(Object object)	For each of the primitive types there is a convenience "get" method; myField.getInt(myObject) is equivalent to ((Integer) myField .get(myObject)).intValue()
void setInt(Object object, int value)	Once again, convenience "set" methods for each primitive type; myField.setInt(myObject, 4) is equivalent to myField.set(myObject, new Integer(4))

TABLE 2 Class Field highlights

You can use Reflection in a variety of ways, but sometimes there are better tools to accomplish the same task. Suppose you want to find out whether an object contains a certain method and, if it does, invoke it. You can do this using Reflection (see Listing 4).

Java has a cleaner way to do it, however. Declare an interface that declares the method and implement that interface in the classes that have the method. Then call the method in the following way:

```
if (anObject instanceof MyInterface) {
// Does anObject implements MyInterface?
 ((MyInterface) anObject).myMethod();        // If
it does, invoke myMethod()
}
```

This code works if you know in advance the classes that will contain myMethod and whether they implement MyInterface. If you don't know, Reflection is the way to go.

you'll have to add it to your map. If there are 10,000 Color constants, the map will be enormous. This is where you can utilize Reflection to analyze the Color class and find its Color constant names and their values. See ColorTools.java in Listing 5 for details. (Because of space considerations, this listing and the corresponding ones for Cases 2 and 3 appear only on the Web at www.sys-con.com/java/sourcec.cfm.)

Case 2

If you develop GUIs in Java, you must be familiar with and probably resigned to using the wordy and annoying anonymous classes to connect component events to their event handlers. Well, there's still hope: by using Reflection you can remove *all* the anonymous classes.

The trick is to use a naming convention (usually called an *idiom* by the scholars) to relate the components to their events and then use Reflection to analyze the class, find the components and event handlers, add event listeners, and invoke the event handlers.

Class getReturnType()	Returns a Class representing the method's return type. Void.class represents the void return type.
Class getDeclaringClass()	Returns the Class object that represents the class or interface that declared the method.
String getName()	Returns the method's name.
Class[] getParameterTypes()	Returns the parameter types for the method. If the method doesn't have parameters, the returned array will have no elements.
Object invoke(Object object, Object[] parameterValues)	Invokes the method on a given object, passing to it the given parameters. If the method is static, the object can be null.

TABLE 3 Class Method highlights

Class getDeclaringClass()	Returns Class object that represents the class or interface that declared the constructor.
Class getParameterTypes()	Returns parameter types for the constructor. If it is the default constructor, returned array will have no elements.
Object newInstance(Object[parameterValues)	Uses the constructor with specified parameter values to create new object instance.

TABLE 4 Class Constructor highlights

Putting Everything Together

Reflection can be used in very different contexts to achieve completely different results. And since you must be eager to see some action at this point, I'll present three cases in which Reflection delivers elegant solutions.

Case 1

Suppose you want to have a function to convert strings into colors. The strings you'd like to pass are color names, and the function should return the appropriate color:

```
Color myColor =
ColorTools.
getColorByName(
"black" );
// myColor will con-
tain the color
Color.black
```

If you don't use Reflection, you have to maintain a mapping structure that relates color names to Color objects. If the folks at Sun decide tomorrow to introduce the pinkPanther Color constant,

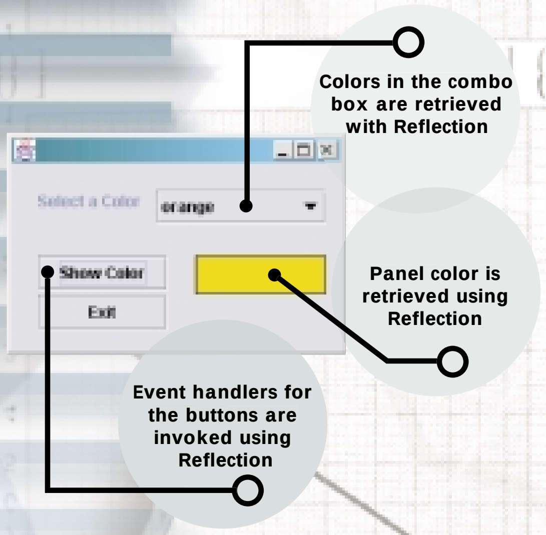

Select a Color
orange
Show Color
Exit
Colors in the combo box are retrieved with Reflection
Panel color is retrieved using Reflection
Event handlers for the buttons are invoked using Reflection

FIGURE 1 Form created by ReflectionTest.java Reflection

Because there's no explicit code in your form that relates handlers to events, the event handlers seem to be called by magic. This is why I like to call this methodology "Magic Couplers." Refer to MagicCoupler.java in Listing 6 to see how to accomplish it – there's no magic after all....

If you adopt this elegant technique, your GUI code will once again be about handling the events, not about connecting event handlers. Magic Couplers have two drawbacks:

1. This one, inherited from the security issues, is the need to declare the components and the event handlers as public so we can access them with Reflection.
2. The connection between events and event handlers is done now at runtime. It means there's no compile-time checking to ensure that the event handlers are named correctly. You can add code (which I removed from the example for brevity) to check that the event handlers correspond to a component and report "unlinked" event handlers.

ReflectionTest.java (see Listing 7) creates a form with a combo box that displays all the color names and two buttons that trigger their event handlers using Magic Couplers (see Figure 1).

Case 3

Another great use of Reflection is for creating application plug-ins. You can design software that allows you and third-party vendors to create extensions for it. This is accomplished very simply. First, define the plug-in interface that enables you to access the plug-in.

```
package plugins;

public interface MyApplicationPlugIn {
    // Interface definition here
    …
}
```

Every plug-in must implement the interface (that's what makes it a plug-in of your application).

"ANOTHER GREAT USE OF REFLECTION IS FOR CREATING APPLICATION PLUG-INS"

```
package plugins;

public class APlugIn implements
MyApplicationPlugIn {
    // Interface implementation here
    ….
}
```

Now you can browse the plug-ins directory to get the plug-in names and dynamically load them by calling:

```
// plugInNames contains the fully qualified
names of the plug-in classes
for ( int i  = 0; i < plugInNames.length;
i++ ) {
    MyApplicationPlugIn plugIn =
(MyApplicationPlugIn)Class.classFor(
plugInNames[i] );
    // Do something with the plug-in here
    ….
}
```

Summary

Java Reflection gives you a metalanguage to ask questions and manipulate classes, interfaces, and objects.

The class Class describes the different attributes of Java classes and interfaces in terms of Field, Method, and Constructor objects. These objects in turn let you inspect and manipulate object attributes and create new objects dynamically.

Now that you know the power and dangers of Java Reflection, use it wisely!

Author Bio

José María Barrera is director of Internet applications development at Caminus Corp., a leading software company for the energy sector. A designer/creator of software using Java and XML, José has been involved with computers for the last 17 years. He holds an MS degree in computer science from New York University.

jose.barrera@caminus.com

Listing 1

```
java.awt.Point p = new java.awt.Point(10,20);
Class pointClass = p.getClass();
// You could call Point.class as well
Field xField = pointClass.getField( "x" );
xField.setInt( p, 3 );
  // Set receives an object as its second parameter,
                                                         // we
have to wrap the 3
System.out.println( p.x );
// Will print 3
```

Listing 2

```
Class[] parameterTypes = new Class[]{String.class};
Object[] parameterValues = new Object[]{"this line was printed
using Reflection"};
// Get the class for System.out
Class class = System.out.getClass();
Method printlnMethod = class.getMethod( "println", parameterTypes );
printlnMethod.invoke( System.out, parameterValues );
// The string is printed here
```

Listing 3

```
Class[] parameterTypes= new Class[]{Integer.TYPE, Integer.TYPE,
  Integer.TYPE};
Object[] parameters = new Object[]{new Integer( 255 ), new
  Integer( 0 ), new Integer( 0 )};

Class colorClass = Class.forName( "java.awt.Color" );
Constructor colorConstructor = colorClass.getConstructor
  ( parameterTypes );

Object myRedColor = colorConstructor.newInstance( parameters );
```

Listing 4

```
try {
// Try to get the method myMethod () from the class of the
      object myObject
 Method myMethod = anObject.getClass().getMethod
   ( "myMethod", new Class[]{} );
// Invoke the method
 myMethod.invoke( myObject, new Object[]{} );
}
catch ( NoSuchMethodException nSchMetE ) {}
// Thrown by getMethod if the method is not found
catch ( SecurityException secE ) {}
// Thrown by the security manager to indicate a security violation
catch ( IllegalAccessException illAccE ) {}
// Thrown if current method does not have access to myMethod
catch ( IllegalArgumentException illArgE ){}
// Thrown if an argument passed is inappropriate
catch ( InvocationTargetException iTarE) {}
// Thrown if myMethod throws an exception, you can call
//  iTarE.getTargetException() to get the exception.
```

Download the Code! www.JavaDevelopersJournal.com

JASON BRIGGS J2ME EDITOR

Cat Fight

If the computer industry was a cat fight, right now fur would be flying in every direction. Microsoft's recent decision to drop Java from their Windows XP distribution is a prime case in point. Spin merchants pop up left, right, and center to fire a barrage of FUD (Fear, Uncertainty, and Doubt) missiles at whomever will listen, and then vanish in a puff of marketing dust. The general media is just as guilty of hyping this mini battle of words – I've lost count of the number of reporters who seem to think that "Java-on-the-PC-desktop" is equivalent to "the-whole-of-Java," and predict doom and gloom for all Java developers if Microsoft doesn't distribute their VM with the operating system.

Meanwhile, it's interesting to note that although a few developers in the community have got caught up in the excitement, the rest of us try to ignore it and get on with the tasks at hand. Programmers keep on programming; innovators keep on innovating.

On the subject of "getting on with things and ignoring it," I recently looked at a forthcoming offering from one company, which will no doubt benefit from users having to employ the plug-in instead of Microsoft's fast but substandard JVM. Lux Inflecta (www.lux-inflecta.com) is a UK/Icelandic company that has come up with a somewhat unique offering in the hosting industry. Rather than just providing an application server and hosting service, GIZA is a full Java development and deployment environment offering an install-on-demand IDE and featuring support for J2EE, J2SE, and J2ME (including MIDlets and Spotlets). Basically a developer can develop his or her application and run it from Lux Inflecta's server, without a huge outlay for their own equipment. GIZAee, their flagship product, has a one month free trial and is also free to academics. Yet another example of how companies are coming up with novel products using Java technology.

The Mobile Game Interoperability Forum, which was in the news recently, has some rather big names as founding members: Ericsson, Motorola, Nokia, and Siemens. According to an information sheet that recently arrived on my desk, the goals of the MGI Forum are to define both an interoperability specification for mobile games and APIs for network-based servers, focus on client/server style wireless games, and help games developers produce and deploy their games.

Quite an ambitious set of targets; so it will be interesting to see what the Forum comes up with over the next year or so.

The subject of games development brings me to this month's issue of ***JDJ*** – Chris Melissinos, Sun's chief gaming officer, is a man dedicated to bringing Java-based games to a machine near you, be it the computer, the games console, or the handheld device. You'll find his thoughts in this month's guest editorial. Following on that theme, Tom Sloper, games producer extraordinaire, presents a treatise to help those who might be involved in designing those games as well as developing them.

I have to admit, games development has been on my mind quite a bit lately. There must be something about seeing all those mobiles that makes me think of tiny, pixelated people jumping across the screen (actually, the idea running through my head is a little more original, but I'm not giving *that* away).

Also in this month's issue Vincent Perrier from WindRiver presents a piece on "Making Java Work in Embedded Devices." Glenn Coates from Vulcan Machines discusses JVM selection and integration, the types of JVMs you might find for embedded devices, and how your company might go about choosing a virtual machine for your product.

• • •

Are you a developer working on a J2ME application? If so, ***JDJ*** wants to hear about it! You won't win any prizes, but you might help motivate your fellow developers, and get a chance to present your thoughts on the oddities of the J2ME platform. E-mail me your thoughts at jasonbriggs@sys-con.com.

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.

J 2 M E   I N D E X

WRITTEN BY CHRIS MELISSINOS

Battle Ogres Everywhere!

Video games are finally entrenched in popular culture and are as widespread a form of entertainment as movies and television. What's most startling is that the games industry achieved this entertainment parity without relying on the standards found in the television and movie industries. This was possible because games were played on proprietary systems designed to do one thing: play games. But in today's world of 206MHz PDAs and 3D-capable cell phones, games are being played everywhere, and for a content industry that has no set standards this presents a significant problem: chasing customers across multiple devices.

Compounding this problem is the onslaught of competition from large media companies that already have broadcast standards in place. They place their content on the same game-playing devices and fight for t4he same consumer's attention. To compete effectively in this arena, game developers need to stop thinking about individual devices as game platforms and consider these devices as part of the landscape encompassed by their games. This is why I strongly believe that Java will emerge as the unifying technology for the games industry.

Java technologies have made tremendous strides in improved performance in the past couple of years, approaching the speed of compiled C++ code today. With Java 1.4 and Java3D, the ability to deliver true cross-platform, high-performance gaming is here today. Consider the first person shooter developed by Full Sail Real World Education, a media school in Orlando, Florida, shown at QuakeCon in August of this year. The game, Jamid, was developed using Java3D and Java 1.3 and runs full-screen at 60+ frames per second on a sub-$1,000 PC. Oh yeah, it also runs *unmodified* on Windows, Solaris, and Linux.

Running such a game on a mobile phone might not be the best use of the phone's capabilities – what with limited memory, input, and performance – so the challenge is in discovering how to incorporate the mobile phone into a game framework. Let's use a massively multiplayer RPG (MMRPG) as an example. In an MMRPG there's a tremendous amount of time that typically goes into playing and maintaining a character. The average EverQuest player, for example, spends an average of 28 hours a week on the game. The problem is that when you're maintaining your character, you're not playing the game.

By moving the less computationally intensive components (such as character maintenance, communication, and trading functions) to a mobile device, you achieve three things.

1. ***Keep your content in front of your customer:*** This keeps the game world, characters, and brand in front of the game player. It's a world where players spend at least one day a week in, and you're enabling them to engage in more pieces of that world.
2. ***Allow the player to prepare characters for game play at a later time:*** Players can do the tedious maintenance separately from the gaming rig, preparing their character from their mobile phone while sitting on the train on the way home from work or during their lunch break. Later, when the players are ready to sit down and battle some ogres, they just jump in and start battling.
3. ***Derive more revenue from your customer:*** If you have someone playing a game 28 hours a week and paying you $10 a month for the privilege of having this addiction, being charged $5 more for access to the game components from any connected device is not a huge leap for the player to make.

The advantages that Java will provide in the form of cross-platform development, reduction in time-to-market, reuse of code, discovery of alternate revenue channels, and providing game companies the ability to reach out beyond the traditional console market and leverage the Web as the game platform, will be unprecedented. There are already significant efforts underway to build the components in Java to make it the de facto standard for game development. Sun Microsystems, Inc., is working with several major game developers and hardware manufacturers, including Sony, on

–continued on page 78

chris.melissinos@sun.com

AUTHOR BIO

Chris Melissinos is Sun Microsystems' chief gaming officer and is responsible for the development of Sun's programs and strategies targeting the electronic entertainment industry.

SYS-CON MEDIA

PUBLISHER, PRESIDENT, AND CEO
FUAT A. KIRCAALI fuat@sys-con.com

ADVERTISING
SENIOR VICE PRESIDENT, SALES AND MARKETING
CARMEN GONZALEZ carmen@sys-con.com
VICE PRESIDENT, SALES AND MARKETING
MILES SILVERMAN miles@sys-con.com
ADVERTISING SALES DIRECTOR
ROBYN FORMA roybn@sys-con.com
ADVERTISING ACCOUNT MANAGER
MEGAN RING megan@sys-con.com
ASSOCIATE SALES MANAGER
CARRIE GEBERT carrieg@sys-con.com
SALES ASSISTANT
ALISA CATALANO alisa@sys-con.com

EDITORIAL
EXECUTIVE EDITOR
M'LOU PINKHAM mpinkham@sys-con.com
EDITOR
NANCY VALENTINE nancy@sys-con.com
MANAGING EDITOR
CHERYL VAN SISE cheryl@sys-con.com
ASSOCIATE EDITOR
JAMIE MATUSOW jamie@sys-con.com
ASSOCIATE EDITOR
GAIL SCHULTZ gail@sys-con.com
ASSOCIATE EDITOR
BRENDA GREENE brenda@sys-con.com
ASSISTANT EDITOR
LIN GOETZ lin@sys-con.com

PRODUCTION
VICE PRESIDENT, PRODUCTION AND DESIGN
JIM MORGAN jim@sys-con.com
ART DIRECTOR
ALEX BOTERO alex@sys-con.com
ASSOCIATE ART DIRECTOR
LOUIS F. CUFFARI louis@sys-con.com
ASSISTANT ART DIRECTOR
CATHRYN BURAK cathyb@sys-con.com
GRAPHIC DESIGNER
ABRAHAM ADDO abraham@sys-con.com
GRAPHIC DESIGNER
RICHARD SILVERBERG richards@sys-con.com
GRAPHIC DESIGNER
AARATHI VENKATARAMAN aarathi@sys-con.com

WEB SERVICES
WEBMASTER
ROBERT DIAMOND robert@sys-con.com
WEB DESIGNER
STEPHEN KILMURRAY stephen@sys-con.com
WEB DESIGNER
PURVA DAVE purva@sys-con.com
WEB DESIGNER INTERN
CAROL AUSLANDER carol@sys-con.com

ACCOUNTING
ASSISTANT CONTROLLER
JUDITH CALNAN judith@sys-con.com
ACCOUNTS RECEIVABLE
JAN BRAIDECH jan@sys-con.com

SYS-CON EVENTS
VICE PRESIDENT, SYS-CON EVENTS
CATHY WALTERS cathyw@sys-con.com
CONFERENCE DIRECTOR
DANIELLE NAPPI danielle@sys-con.com
CONFERENCE MANAGER
MICHAEL LYNCH mike@sys-con.com
SALES EXECUTIVE, EXHIBITS
MICHAEL PESICK michael@sys-con.com
SALES EXECUTIVE, EXHIBITS
RICHARD ANDERSON richard@sys-con.com
SHOW ASSISTANT
NIKI PANAGOPOULOS niki@sys-con.com
JDJSTORE.COM
ANTHONY D. SPITZER tony@sys-con.com

J2ME FAQ

IS PERSONALJAVA PART OF J2ME?

The short answer is yes. For the long answer, we'll refer to Sun's FAQ for J2ME, which states that PersonalJava was the "first Micro Edition technology." Because PersonalJava has been around for a while now, you'll find more products with a version of it installed. But sometime this year (2001), Sun is expected to replace the existing PersonalJava technology – based on Java 1.1 – with a new release based upon Java 2, and incorporated into the J2ME concepts of Configuration and Profile components.

IS ALL THE JAVA API WITHIN J2ME?

No. Even PersonalJava – which has the most complete coverage of the Standard Edition API – is still just a subset.

WHAT IS A "MIDLET"?

Actually, the correct word is MIDlet. A MIDlet is an application written for MIDP (the Mobile Information Device Profile). You might find these on mobile phones, PDAs – in general, small devices.

CAN I USE THREADS? IS THERE A PENALTY?

Yes, you can use threads, unless you're writing a JavaCard applet. As for the penalties, it very much depends upon how you want to use them, and the environment you are working within. When developing for constrained devices, you always have to keep the resources you have available in the back of your mind. If you're writing a MIDlet, and create 100 threads to try to load 100 images simultaneously, then there definitely will be a penalty – it undoubtedly won't work.

DO I USE AWT OR SWING FOR MY GUI?

If you're developing a PersonalJava application, then you have access to a modified version of AWT – "modified" meaning that a few java.awt classes/methods are optional, that some have been changed, and that there are some additions to the basic package. You may be able to get Swing to work within a PersonalJava environment as well. A brief skim of the PersonalJava forums show some success stories – and more than a few painful attempts. None of the other J2ME "products" support AWT or Swing (for example, MIDP has the javax.microedition.lcdui package, for user interfaces).

WHERE CAN I FIND MORE INFORMATION ABOUT WIRELESS TECHNOLOGIES?

The back issues of *JDJ* are one place you can look. For online information, you can look at the following URLs:

1. http://developer.java.sun.com/developer/products/wireless/
2. *Bill Day's J2ME archive:* www.billday.com/j2me/
3. *Sun's Wireless forums:* http://forum.java.sun.com/

WHERE CAN I DOWNLOAD J2ME EMULATORS?

The J2ME Wireless Toolkit: http://java.sun.com/products/j2mewtoolkit/download.html

To download the MIDP reference implementation on this page: http://java.sun.com/products/midp/

CLDC : www.sun.com/software/communitysource/j2me/cldc/download.html

CDC (and the Foundation profile): www.sun.com/software/communitysource/j2me/cdc/download.html

WHERE CAN I FIND DEVICES THAT RUN J2ME?

Move to another country. At the moment, there are a limited number of countries where J2ME capable devices have been released – especially for mobile phones. While you can probably find PDAs that support PersonalJava almost anywhere in the world, the same is not true for mobiles.

In Japan, NTT DoCoMo has a number of phones from Panasonic, Fujitsu, Sony, and others (only available in Japan, of course). In the U.S., Motorola has a couple of J2ME capable mobiles. For a more comprehensive list, check out www.javamobiles.com/

Together We Stand, Divided We Fall

–continued from page 5

cases is far removed from Java.

We as a community are entering some troubling times. Our decision to run with Java is going to be questioned again by clients and upper management. Microsoft is doing all it can to arm these computer-illiterate people with the information to fight Java, and to the unthinking it does sound pretty convincing.

Our solutions will have to work twice as hard to weather the storms we're about to encounter. We know the language doesn't need to prove itself, but the solutions we labor in will have to stand up to a severe beating from Redmond. To that end, we have to look at the bigger picture and take responsibility for more than just our "cog" in the engine.

Thinking Outside the Box ...

–continued from page 12

applications executing on BEA's application servers to connect to EIS resources through JCA connectivity to WebMethod's adapters. It's obvious that both companies compete in the application-execution space; however, it's testimony to the promise of JCA that they chose to partner in the connectivity arena.

The J2EE platform allows us to properly define the boundaries of our application box, but the actual definition depends on the environment, your company's core competency, and your faith in the J2EE vision.

Battle Ogres Everywhere!

–continued from page 76

the Java Game Profile (JSR-134). This Java Specification Request identifies the various areas of game development, and the participating game developers are working together to build the components and technologies that will enable the first true cross-platform game development technologies. Sun has also launched www.JavaGaming.org to act as ground zero for Java game development, offering everything from discussion boards to sample code to cool links.

As the pressures and costs of developing increasingly compelling game content in the face of new competition continue to rise, the benefits of Java become more attractive. With the high level of penetration across dozens of different media devices, Java is providing an exciting, high-performance platform for the next generation of games. As game developers learn how to incorporate these devices into their game framework, and look beyond the box to the Web as the platform, the ubiquity of Java will prove invaluable. Of all the great services and technologies that are incorporating Java, none will experience a greater impact than the games industry. So check your mobile phones, PDAs, cable boxes, and game consoles. Your favorite game will show up where you may least expect it: everywhere.

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: 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 on 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 envronment

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

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 and 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 Java TV page at the following: 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: 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/

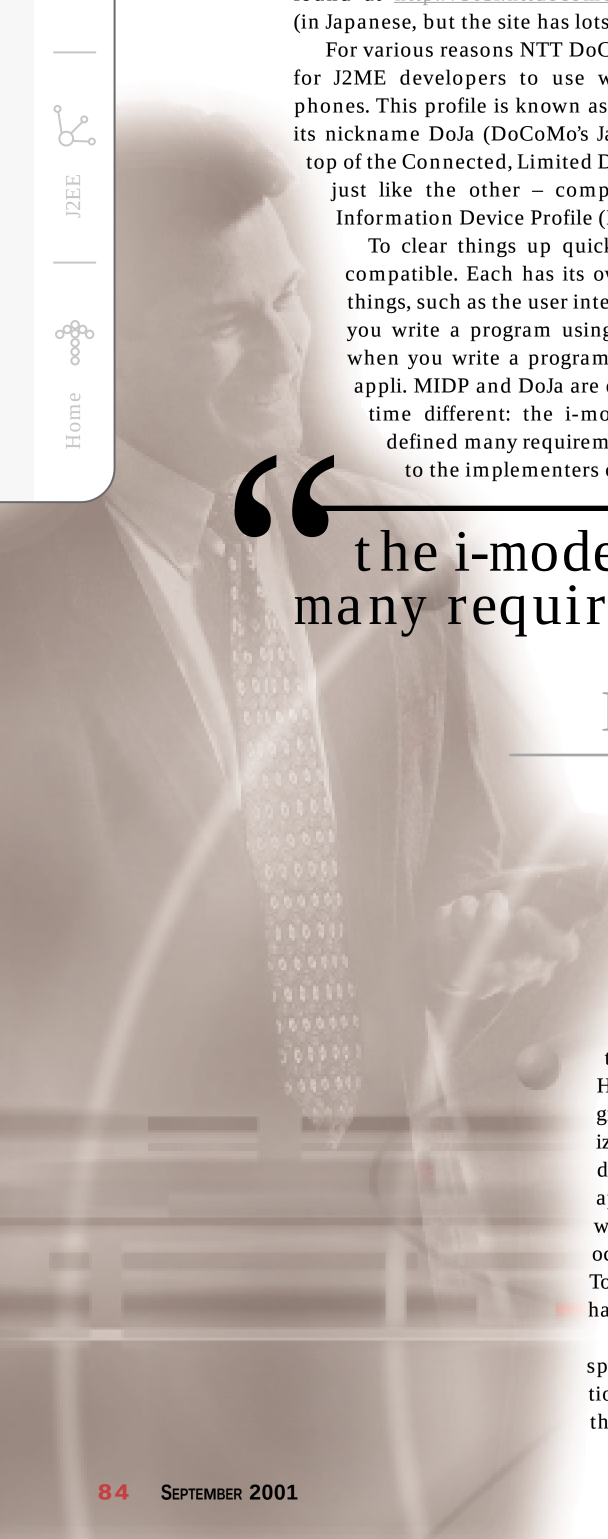
DoJa in NTT DoCoMo Phones

A few basics on developing i-mode Java in mobile phones

Written by Zev Blut

Here in Japan, where it's common to see someone thumbing away on a cell phone while on the train, NTT DoCoMo was one of the first companies in the world to release Java-capable mobile phones. In the ensuing six months, more than four million users have adopted the phones – and the number is growing. That's a good enough reason for an aspiring J2ME programmer with a sense of adventure to learn a little about programming Java applications for NTT DoCoMo phones.

"NTT DoCoMo released its own profile for J2ME developers, known as i-mode Java"



Before I go into the technical details on how to develop for these phones, let's talk about the types of devices available today. Six models of the 503i series mobile phone, provided by five manufacturers, are currently available to the consumer .

A few common hardware traits of these mobile phones:

- They are small, light, and provide long hours of talk time.
- They provide a color screen, with at least 16 harmony audio capability for ringtones and for use in applications.
- They have the capability to display GIF images.
- The phones can also write e-mail, browse a reduced form of the Web, and download backgrounds and ringtones for customizing the phone and, of course, Java applications.

More information about the 503i series phones can be found at http://503i.nttdocomo.co.jp/normal/n_index.html (in Japanese, but the site has lots of pictures).

For various reasons NTT DoCoMo released its own profile for J2ME developers to use when programming for the phones. This profile is known as i-mode Java – also called by its nickname DoJa (DoCoMo's Java). I-mode Java resides on top of the Connected, Limited Device Configuration (CLDC), just like the other – competing – profile, the Mobile Information Device Profile (MIDP).

To clear things up quickly, MIDP and DoJa are not compatible. Each has its own API and way of handling things, such as the user interface and data storage. When you write a program using MIDP, you write a MIDlet; when you write a program using DoJa, you write an i-appli. MIDP and DoJa are quite similar, but at the same time different: the i-mode Java specifications have defined many requirements that are currently left up to the implementers of MIDP devices to decide.

"the i-mode Java specifications have defined many requirements that are currently left up to the implementers of MIDP devices to decide"

I-Mode Java Specifications

When I first looked at the i-mode Java specifications, my reaction was that it's too strict. DoJa specifies that all applications must be less than or equal to 10KB in size. How can you get a great program that is going to revolutionize the world to fit in 10KB? Well, don't fret. Many interesting applications have already been written and keep me happily occupied on my train rides to Tokyo. I'm surprised by what I have been able to fit in 10KB.

Another requirement of the specification is that applications must be downloaded to the handset from a Web page, thus solving the over-the-air issues that some have had with MIDP phones.

All applications have the ability to store up to 5KB of permanent data on the phone. Unlike MIDP, where you have MIDlet Suites that can share data in storage, DoJa doesn't allow for the sharing of data among i-applis.

As mentioned previously, all phones must support the GIF image format. In addition, all phones must allow HTTP/HTTPS (SSL) connections to the host server that the i-appli was downloaded from (SSL is built into the phones, providing end-to-end security from the handset to the Web site, thus easily providing a solution for secure transmissions).

Not directly related to the specifications but still important to know is that NTT DoCoMo charges the user 0.3 yen for every 128 bytes of information sent from and received by the phone. If you want happy users, you need to keep their bills in mind when writing network-intensive applications.

DoJa provides four packages that are added to the ones found in the CLDC: com.nttdocomo.io, com.nttdocomo.net, com.nttdocomo.ui, and com.nttdocomo.util. Later in this article, I'll use some of the classes in the com.nttdocomo.ui package to help you get acquainted with developing i-applis.

Resources and Tools for Developing I-Applis

Now that I've introduced the basics, let's cover what you need to do to start developing i-applis. First, go to NTT DoCoMo's Web site about i-mode Java and download two English .pdf files (www.nttdocomo.com/i/java/index.html). These documents are an API listing and a developer's guide to creating i-applis. They'll be of considerable use for further exploration.

Once you have these documents, you need to get an emulator and an SDK to develop with. First you should get the CLDC SDK from http://java.sun.com/products/cldc/. This provides the CLDC base classes, source code, and documentation.

If you're familiar with the J2ME Wireless Toolkit provided by Sun Microsystems for developing MIDlets, then you'll be pleased to know that NTT DoCoMo provides a modified version of this toolkit for developers. It can be found on the Japanese Web site at http://www.nttdocomo.co.jp/i/java/tool.html.

If you can't read Japanese and are having trouble with the page, don't worry. Take it through a machine translation site, such as AltaVista's Babel Fish at http://babelfish.altavista.com). The toolkit will run in English (if you aren't running a Japanese OS) and provides an emulator to run your i-applis on. As a bonus, it also automates many of the compilation, preverification, and packaging steps discussed later.

Another emulator, probably new to developers from the MIDP world, is Zentek Technology's i-JADE product (found at www.zentek.com/i-JADE/index.html). It's designed to emulate the look and feel of actual phones available in Japan right now. These emulators are invaluable tools for developers, and I suggest you download both.

Finally, you'll need a J2SE SDK for use during development. You should have no problems using most IDEs with DoJa, but if you want to integrate development into the IDE, then Forte for Java, JBuilder, and VisualCafé all offer ways to integrate the emulators.

Writing an I-Appli

Now that you know where to find resources and tools for i-mode Java, we're ready to write a simple application. For this article it's going to be the tried-and-true Hello World, found in Listing 1.

The first thing to note is that all applications must have a class that extends com.nttdocomo.ui.IApplication. This class must implement IApplication's start method, which is first called when the user runs the application on the phone. This is similar to an applet's start method or a MIDlet's startApp method. In the start method I create a Panel (a class that represents a screen and lets you place objects on it), called *hpanel,* then make a Label, called *hlabel,* with the text "Hello World!" and add it to the panel I created.

Next, I set hpanel's softkey 2 text to show "quit" and register the HelloWorldAppli class as the SoftKeyListener. All phones have two buttons called *softkeys* located on the left and right edges, below the handset's screen. When you set the text for a softkey, it will be displayed directly above the button.

Softkeys are commonly used as a way to quickly activate functions or menu options. In this example pressing the right button (softkey 2) will quit the application. One point to remember is that all i-applis must quit by calling the IApplication terminate method. Don't call System.exit() or Runtime.exit() as they will cause a SecurityException.

We catch that a softkey has been pressed by making our class implement SoftKeyListener with the two methods softKeyPressed and softKeyReleased and register the listener to the panel to catch the events when the panel is showing.

Finally, we're ready to show the message to the user. Thus we call the static setCurrent method on the singleton class Display (passing in hpanel). This is a rather long and involved version of Hello World, but if we did a simpler version, using System.out instead of creating Panels and Labels, the message would never be shown on a real phone, because the phones don't show System.out or error messages.

Of course, I could decide not to bother to implement the quit command, but in that case the user would then have to press the force quit key (usually the power button). Pressing this button drops the user out of the i-appli software listing and back to the base screen the phones show when powered on. If they pressed our softkey instead, it would quit the application and return to the listing of downloaded i-applis on the handset, which is much more user-friendly.

Figures 1–4 are screenshots of the application running on the wireless toolkit, i-JADE, and on a real phone.

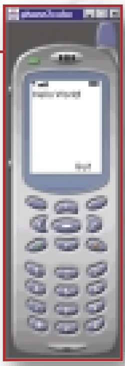


FIGURE 1 Hello World on the J2ME Wireless Toolkit for DoJa

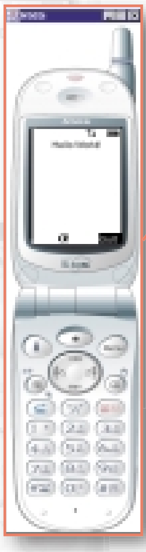


FIGURE 2 HelloWorld on i-JADE emulating the F503i phone

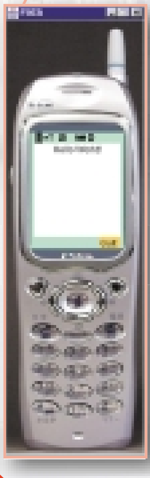


FIGURE 3 HelloWorld on i-JADE emulating the N503i phone

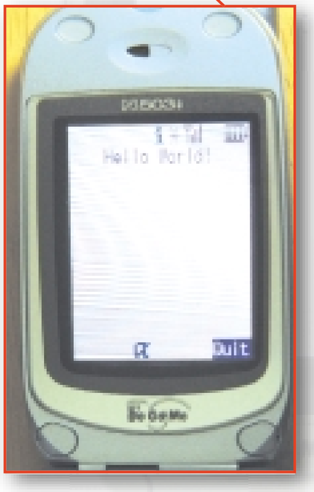


FIGURE 4 HelloWorld on a real N503i phone

Compiling I-Appli

The steps needed to go from code to pixels shown on a real phone are fairly similar to those needed for an MIDP application, but a bit more involved for a J2SE application. In all cases, once you've written your code you must compile it. A trusty J2SE compiler will do the job just fine.

When compiling you must set the bootclasspath parameter to the CLDC base library instead of the usual J2SE base, then set the classpath to point to your source code and the DoJa API. Make sure to turn off debugging info, because it'll add to the size of your application and be of no use while running on a real phone. In this example our compilation command might look like this:

```
c:\jdk1.3\bin\javac -g:none –bootclasspath
c:\j2me_cldc\bin\api\classes –classpath c:\ijadeP\i-
jade-p.jar;. -d compiled HelloWorldAppli.java
```

Run Preverify

Next you must preverify your compiled code. For those who do this on a daily basis with MIDP, it's exactly the same (except for a few different parameters). For those who don't know, "preverify" is a tool used to offload some of the work the KVM must do to ensure that the Java bytecodes it receives are valid. It's included in the CLDC SDK, and it can also be found in the wireless toolkit for DoJa.

Following is an example of running the preverify command on the HelloWorldAppli class:

```
c:\j2me_cldc\bin\preverify -classpath
c:\j2me_cldc\bin\api\classes;c:\ijadeP\expandedJar\ -d
preverified compiled
```

To explain, I run the preverify command, setting the classpath to the CLDC's classes and the DoJa classes, and take the classes in the compiled directory, outputting the resulting preverified classes to a folder called *preverified*. In this example I'm using i-JADE for the DoJa API – the preverify tool can't read .jar files, so you must expand any .jars you're using, setting the classpath to point to the expanded folder (in this case the folder used is expandedJar).

Packaging I-Appli

After running preverify, you can package the application to deploy it. I-mode Java phones take a .jar file with the preverified classes and any other resources you wish to use, such as GIFs and sound files. Make use of the J2SE's .jar tool to bundle your application into a .jar file. When making a .jar, the handsets won't use the Manifest file; therefore you can save space by using the flag *–M* in your .jar command.

As in the following example:

```
c:\jdk1.3\bin\jar -cvfM HelloWorld.jar
HelloWorldAppli.class
```

Once you've made your application a .jar, look at the file. If it's less than 10,240 bytes, you're in the clear and ready to take the next step in deploying your application. If it's greater than 10KB, trim your classes and/or resources so your .jar file is 10KB.

Creating an Application Descriptor File

Once we have the .jar file, we're ready to create an Application Descriptor File, otherwise known as a .jam file. This file is used to tell the phone about your application before you download and execute it. It contains vital information, such as the application size and name, parameters to pass to the application, and whether it uses the network. For this article I'll introduce you only to the details that are mandatory for all applications: AppName, AppClass, AppSize, PackageURL, and LastModified.

- ***AppName:*** Name of the application; used when showing the list of the applications on the handset.
- ***AppClass:*** Class that extends IApplication. If this class is part of a package, make sure to give the fully qualified class name.
- ***AppSize:*** Size of the .jar file containing the i-appli in bytes.
- ***PackageURL:*** URL pointing to the .jar file to download. If the .jam and .jar file are in the same directory, then just give the name of the .jar file.
- ***LastModified*:** Time the application was last updated. Used to check if upgrades are available. This is also the trickiest key to work with. If the next five steps aren't followed, it can lead to problems at download time.
  1. The first three letters of the day of the week followed by a comma
  2. The two digit date
  3. The first three letters of the month
  4. The four digit year
  5. Finally, the time up to the number of seconds, separated by colons and using two digits for each number

It can be a pain, but with practice it becomes easy (even Panasonic made a mistake implementing this spec, so beware of the month of April if you want to have your app run on a Panasonic phone). Listing 2 is an example .jam file for the HelloWorld i-appli named HelloWorld.jam.

Note that the wireless toolkit and i-JADE Lite Plus can automate the compilation, preverification, and packaging process, and can generate the .jam file, greatly simplifying development and saving you from the vagaries of the LastModified key.

Developing a Web Page for Deployment

The final steps involve creating a special Web page. Upload this page along with the .jar and .jam files for the application onto a Web server. To create the page, follow NTT DoCoMo's modified cHTML format. This is HTML with fewer frills; a good site for information about the tags is http://www.nttdocomo.com/i/tag/lineup.html). The Web page must include two specific tags: an Object tag and an Anchor tag. Listing 3 provides a sample Web page for the application, and Figure 5 shows what it looks like on a real phone.

To make the Web page, create an object tag that specifies an ID to refer to the anchor tag and a data attribute that gives the path to the .jam file. The object tag also has a type attribute with the value "application/x-jam".

For the anchor tag insert a special attribute *ijam* that references the object tag's ID value. Then give an HREF attribute pointing to a page to display if the user attempts to download the i-appli from a handset or browser that doesn't support i-applis.

"For those who don't know, "preverify" is a tool used to offload some of the work the KVM must do to ensure that the Java bytecodes it receives are valid"

Once you've created the Web page, all that's left is to upload the page along with the .jar and .jam files to a Web site. If you're in Japan, open up the page in your phone and click on the anchor to download and run the application.

Summary

You've now seen what it takes to make a simple application for NTT DoCoMo's Java-enabled handsets. It might seem like the various steps to develop and release an i-appli are a chore, but practice makes it easy, and the time-consuming task eventually returns to writing the code itself. Now it's time to come up with more of those great applications that keep everybody so occupied that they miss their intended destination on the train!

Resources

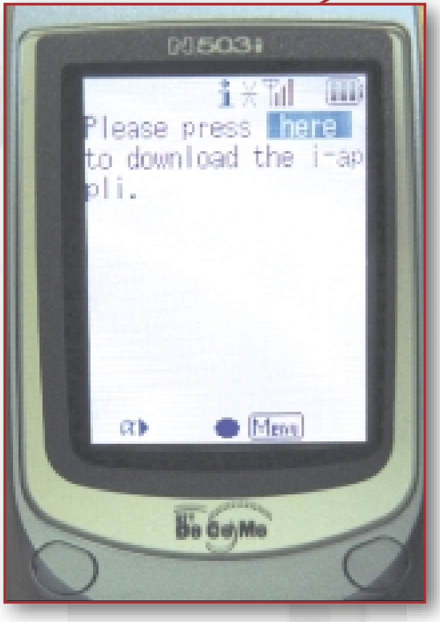
1. *Mailing list focusing on the mobile phones in Japan:* www.appelsiini.net/keitai-l/
2. *NTT DoCoMo's Web site about i-mode:* www.nttdocomo.com/i/index.html
3. *NTT DoCoMo's listing of the Java-enabled 503i series phones:* http://503i.nttdocomo.co.jp/
4. *Web page I created to help beginners get started developing i-applis:* www.geocities.co.jp/SiliconValley-Cupertino/1621/
5. *Zentek Technology's i-JADE emulator:* www.zentek.com/i-JADE/index.html
6. *Sun Microsystems CLDC SDK:* http://java.sun.com/products/cldc/

Author Bio

Zev Blut is a software developer in the R&D department for Yamatake Corporation in Fujisawa, Japan. He has worked with the i-mode Java API since the phones were released to the public in January 2001. Blut holds a computer science degree from the University of Texas at Austin.

zev@h7.dion.ne.jp

FIGURE 5 hw.html on the N503i phone


Please press here
to download the i-appli.

Listing 1: Source Code for HelloWorldAppli.java

```
import com.nttdocomo.ui.*;

public class HelloWorldAppli extends IApplication implements
SoftKeyListener {

    public void start(){
        Panel hpanel = new Panel();
        Label hlabel = new Label("Hello World!");
        hpanel.add(hlabel);

        hpanel.setSoftLabel(Frame.SOFT_KEY_2,"Quit");
        hpanel.setSoftKeyListener(this);

        Display.setCurrent(hpanel);
    }

    public void softKeyPressed(int softKey){
    }

    public void softKeyReleased(int softKey){
        if (softKey == Frame.SOFT_KEY_2){
            this.terminate();
        }
    }
}
```

Listing 2: Application Descriptor File: HelloWorld.jam

```
AppName = HelloWorld
AppClass = HelloWorldAppli
AppSize = 629
PackageURL = HelloWorld.jar
LastModified = Sat, 04 Aug 2001 23:00:10
```

Listing 3: hw.html: Example Web Page for the HelloWorld I-Appli

```
<html>
<head><title>Hello World i-appli</title></head>
<body>
<object declare id="helloworld.dec" data="helloworldiappli.jam"
type="application/x-jam"></object>
Please press
<a ijam="#helloworld.dec" href="handsetError.html"> here </a>
to download the i-appli.
</body>
</html>
```

Download the Code!
www.JavaDevelopersJournal.com

Making Java Work in Embedded Devices

Leverage business from a new perspective

Part 1 of 2

WRITTEN BY
VINCENT PERRIER

This is the first in a two-part series on the benefits of using the Java development and runtime environment for embedded computing. Java, with its "write once, run anywhere" paradigm, is ideal for embedded computing because of its portability, reliability, security, and Internet capabilities.

However, Java can present some challenges for embedded-systems developers. First, speed – Java applications are inherently slower than applications compiled into native machine code and embedded processors are generally less powerful than those found on desktops. There's no direct memory access and no interrupt handling, two features usually required for developing low-level software to control hardware, and Java can't easily run in real time, to name only a few of the issues.

Despite its limitations, Java can be used effectively for embedded systems. In this article, I share Wind River's experience in embedded real-time computing to help developers overcome some of the common pitfalls inherent in the Java development environment. Part 2 will help you determine if Java is right for your embedded development project.

Java for the Embedded World

Unlike desktop systems, embedded systems use different user interface technologies; have significantly smaller memories and screen sizes; use a wide variety of embedded processors; and have tight constraints on power consumption, user response time, and physical space. The original Java Developer's Kit (JDK) technology was designed for desktop environments with powerful processors, large disks, and large available memory spaces. Consequently, the full JDK architecture is not suitable for many applications in the embedded world.

However, Sun has also introduced versions of the first iteration of Java (JDK 1.1 or Java 1) for the embedded world, namely EmbeddedJava and PersonalJava. The newest iteration, SDK 1.2 and higher (1.3, 1.4) or Java 2, is grouped into three editions, one of which is the Java 2 Micro Edition (J2ME) aimed at the consumer electronics and embedded market. EmbeddedJava, PersonalJava, and J2ME provide standard, platform-independent Java development environments that reduce costs and shrink development cycles for Java applications and applets running on embedded devices.

EmbeddedJava

EmbeddedJava is a scalable and configurable environment suitable for low-end embedded devices with dedicated functionality and limited memory. It's ideal for closed system devices that don't require Web browsing capabilities and don't expose application programming interfaces (APIs) to the outside world. EmbeddedJava includes tools that allow developers to configure and compile runtime environments that contain only those fields and methods necessary for a particular application's needs.

An executable image of a complete EmbeddedJava environment can be generated and placed in the embedded system's ROM. A dedicated tool chain creates optimized application executables known as *ROMlets*, which can be programmed into the device's ROM, and *patchlets*, enhanced ROMlets that can be upgraded in the field. Developers can use EmbeddedJava for a variety of products, including process controllers, instrumentation, office printers and peripherals, and networking routers and switches.

PersonalJava

PersonalJava is an upward-compatible subset of Java dedicated to consumer and embedded devices, and specifically designed for building network-connectable consumer devices for home, office, and mobile use. It consists of the JVM and a subset of the JDK 1.1 APIs, including core and optional APIs and class libraries. PersonalJava includes the specific tools and APIs required by embedded applications in resource-limited environments. Examples of devices suitable for the PersonalJava application environment include mobile handheld devices, set-top boxes, game consoles, and smartphones.

J2ME

J2ME, designed for the development of such devices as digital cellular phones, pagers, personal digital assistants, digital set-top boxes, and retail payment terminals, defines vertical platforms called *profiles* that sit on top of two different configurations.

The connected device configuration (CDC) uses a 32-bit standard JVM and requires more than 2MB of memory. This configuration relies on some kind of connection to a network and on an underlying RTOS and C runtime environment. The connected limited device configuration (CLDC) uses the 16- or 32-bit KVM and requires 256–512KB of memory. It doesn't necessarily require a "persistent" network connection. Profiles within these configurations are integrated into the J2ME framework with each profile targeting a precise vertical market.

As discussed, some mechanisms of the JVM and Java as defined in the desktop-oriented platforms (JDK 1.1 or Java 2) are not suitable for embedded systems. PersonalJava, EmbeddedJava, and J2ME define a framework for optimized embedded Java implementations, but these Java versions alone are not ideal unless coupled with some background and experience in the development of real-time and embedded applications. The key to success is in knowing how to architect Java technology specifically for an embedded device.

Increasing Speed

Java is a semicompiled language and therefore inherently much slower than native machine-code execution. This speed issue is exacerbated by the fact that embedded processors are usually less powerful than desktop ones. The easiest and most expensive way to enhance Java performance is to use a faster processor. Fortunately, there are other solutions.

Just-In-Time (JIT) compilers for desktop JVMs accelerate the Java interpretation cycle by translating Java byte code into machine code on the fly. In their original form, JIT compilers are not suitable for use with embedded applications because they require a lot of dynamic memory. The compilers are also unable to reach the performance of traditionally compiled C/C++ code because they translate Java byte code into native code at runtime as opposed to buildtime. Developers can achieve a tradeoff between performance and memory footprint with dynamic adaptive compilers – JIT compilers customized for embedded applications – that perform statistical analysis of byte code prior to its translation into native machine code.

Flash (or "pass-through" JIT) compilers aren't embedded in the JVM like JIT compilers, instead they run separately on a network host as "compiling" servers. Upon a class download demand, the flash compiler compiles the requested Java bytecode and passes the resulting native code to the JVM. Flash compilers are still runtime compilers and as such, they'll slow runtime execution to some extent. Plus, they deliver classes over a network during application execution, which can also slow execution speed.

Ahead-of-time compilation translates Java source code to C code (losing the Java portability) or translates Java byte code to native machine code (retaining portability at the application level). Unlike JIT or flash compilers, ahead-of-time compilers work at compile-time and can achieve optimizations similar to those achieved by traditional compilers. Developers can avoid undesirable code expansion due to byte code-to-machine-code translation by compiling 20% of the most relevant Java code into native machine code, with the JVM interpreting the remaining 80%.

Executing Code from ROM

Desktop JVMs usually can't execute Java code directly from ROM. Normally, Java classes are first loaded into RAM, verified, and then executed by the JVM. This approach is impractical for many embedded systems because it increases the use of expensive RAM beyond the cost constraints of the embedded system.

A ROMizer, such as Java CodeCompact for PersonalJava and EmbeddedJava, creates ROM-based executable images of Java classes. To execute Java code out of ROM instead of RAM, a ROMizer utility processes Java class files into a runtime format that can be run directly out of ROM or flash memory by the JVM. The ROMizing process frees the JVM from the class-file loading and byte code–verification phases, and improves the performance and start-up time of Java applications.

Direct Access to Memory/Files

Java includes a large set of classes that are not scalable by default. Desktop JVMs usually require several megabytes of disk space and RAM to execute. Embedded systems usually don't have large disk drives or big memory spaces, although some of this memory usage may be converted into ROM/flash memory for diskless embedded systems. Therefore, it's critically important that developers tailor the operating system and runtime components for embedded systems to each application to avoid unnecessary memory usage.

However, it may not be easy to analyze and remove all unused classes and methods from Java to minimize the embedded application's final memory footprint. Desktop JVMs usually download class files from a local hard drive or a file system on the network, and these resources aren't always available to embedded systems.

The best scalability that can be achieved starts with the underlying real-time operating system (RTOS) scalability and can be tracked at three different levels on the Java side. First, the JVM may be scalable depending on the services the application requires. Second, the Java classes may also be sorted and only included in the runtime system if they're used by the application. The verbose option of the Java launcher can be helpful in the analysis of the used classes. Finally, a dedicated utility tool, such as EmbeddedJava's Java Filter, can be used to skim Java classes and remove unused methods and fields.

Developers may want to avoid loading Java class files from a local or networked file system, or use them only as an option. They can either be stored in a memory-based virtual file system (RAM, ROM, or flash) or can be converted with a "file-izer" utility tool into a C data structure that stores the class file byte-code content that's linked to the RTOS image.

Running Java in Real Time

Java's biggest problem with real-time execution is that Java garbage-collection algorithms are usually nondeterministic. Once garbage collection starts, it must run to completion and can't be preempted by a more urgent Java thread. The time required for the garbage collector to run is not predictable. This characteristic creates unbounded latency in event response, a condition that can't be allowed in real-time systems. However, even a deterministic garbage collector may not guarantee a real-time execution of Java programs.

There are currently no real-time Java implementations compatible with standard Java platform specifications, even though a consortium of companies, including Wind River, have been working with Sun to define standard real-time extensions to Java under the Java Community Process (in the JSR-00001, see www.rtj.org). The best existing solution to Java's real-time shortcomings consists of a hybrid solution. Developers can write the real-time part of an application in C/C++ for the targeted RTOS, and the rest in Java using JNI to connect the two worlds.

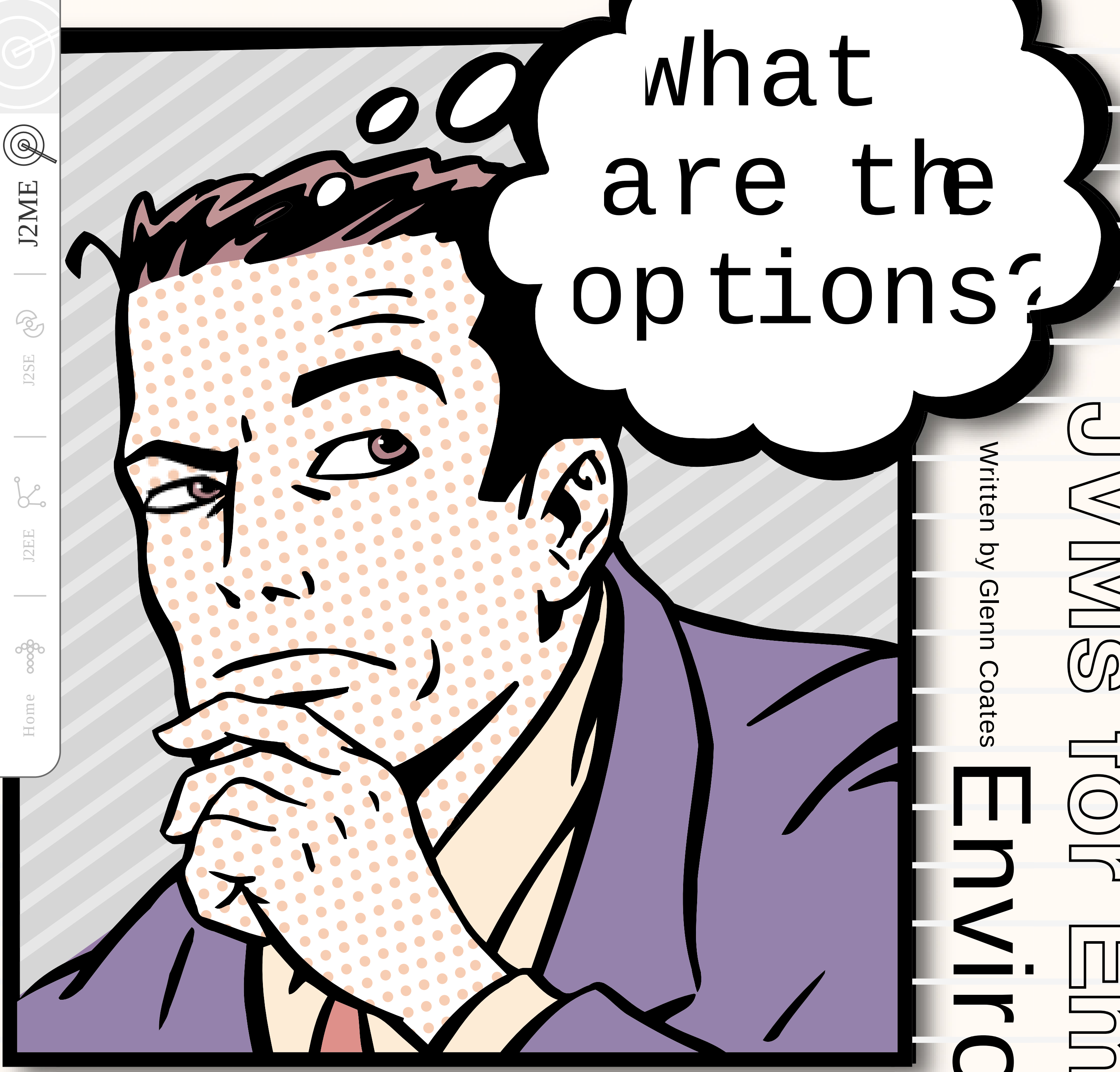
Developers must pay careful attention to the implementation of the JVM on top of the RTOS, making sure the overall system operates in real time even when Java is running. In addition, when evaluating JVM solutions, developers must consider memory management, garbage collection and object finalization, multithreading, interthread synchronization, networking, and graphics.

Conclusion

Java is suited for embedded applications because it can help developers leverage business from a new perspective, especially within the Internet-access device market where connectivity, interactivity, reliability, security, and dynamic extensibility are vital requirements. Embedded devices usually have severe memory and power consumption constraints, lower processor power, real-time behavior, and other requirements that Java doesn't usually handle well. Numerous companies offer leadership products and solutions that can help developers make Java work in embedded solutions.

vincent.perrier@windriver.com

AUTHOR BIO
Vincent Perrier is Wind River's product manager for Java platforms. He has a computer science and engineering degree from the University of Nantes in France.



JVMs for Embedded Environments

Written by Glenn Coates

While representing my company at JavaOne this year, it was apparent that many Java engineers are becoming more interested in the issues surrounding JVM selection and integration. Many questions were asked concerning the trade-offs involved in the different ways of implementing the JVM. This article is aimed at helping device manufacturers, OEMs, and J2ME application engineers understand the issues – and at helping to initiate further questions when talking to JVM vendors.

What Are the 'Risks' Associated with Java?

In my previous article "Java Thick Clients with J2ME" (***JDJ,*** Vol. 6, issue 6), I outlined a number of risks commonly associated with Java. These include:

- Speed
- Memory requirements
- Power consumption
- Licensing and silicon cost

While discussing various Java risks and virtual machine-implementation approaches, I'll refer to the following terms: *code bloat, compile stalls,* and *profiling stalls.*

Code Bloat

This is a real problem when an application's Java byte code is compiled down to the native processor instructions, and is especially the case when using RISC processors. Java byte code is abstract, and does a lot more work in less space than long-winded native instructions. Whenever a Java program is compiled on or off the device, the native code needs to reside in memory so it can be run. This expansion in memory is referred to as code bloat.

Compile Stalls

Compile stalls occur when, for example, a JIT compiler compiles the application's Java byte code into native machine code on the device as the application is running. Obviously, this requires processor power and can result in a noticeable stall as the compiler runs. Depending on the device, processor, clock speed, VM, and the application, this stall may often go unnoticed by the end user if it's "hidden" well enough.

Profiling Stalls

In order to tackle the problems associated with code bloat and compile stalls, some VM vendors decide to compile only the parts of the application that take the longest to execute. However, no matter which technique is used, this profiling will require some processing power, and again may not be noticeable depending on the device, processor and VM, and the application.

It's worth remembering that any of this extra processing can be easily hidden from the user. Simply increasing the clock speed from say 25MHz to 200MHz will do the trick, but the battery power of the device will last for only a day instead of a week!

What Are The Different Types of VMs?

The alternative approaches to VM implementation can be plotted on a time line illustrating the evolution of VMs (see Figure 1):

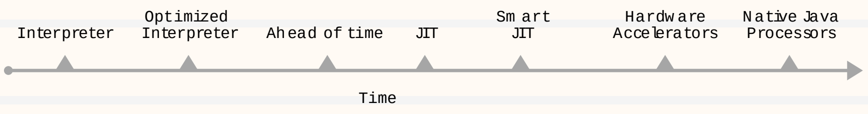


FIGURE 1 Time line

Interpreter

Many JVMs employ pure software interpretation. A good example of this is the SUN KVM. Table 1 outlines its major characteristics.

Speed	About as slow as it gets due to software interpretation overhead. This is even the case on a desktop PC.
Memory Requirements	No code bloat, as the Java byte codes are never compiled into native processor instructions.
Power Consumption	As the applications are interpreted, this will require more processing power.
Licensing and Silicon Cost	Cost of licensing the software VM.

TABLE 1 Interpreter

Optimized Interpreter

When performance becomes a problem, the interpreters can be optimized in order to improve performance (see Table 2). Many devices that use a pure interpreter now use this method as a way of squeezing out that extra performance, and it does have a significant effect.

The optimized software must exist for your hardware, OS, and JVM.

Speed	Will noticeably improve the performance of the pure interpreter, which can help make the application more responsive and usable.
Memory Requirements	Similar to pure interpreter. May need extra static memory to store extra code modules.
Power Consumption	Similar to pure interpreter.
Licensing and Silicon Cost	Extra licensing costs for the software bolt-on.

TABLE 2 Optimized Interpreter

JIT

The Just-in-Time (JIT) approach to Java execution does as its name suggests. It compiles the Java byte code into native processor code as the application is run.

A common misconception with JIT is that the result is an application that will run as quickly as an equivalent native C application once the compilation is complete. This isn't the case as the application is still a Java application and still runs within the Java domain.

Depending on the implementation, it may compile the byte code at different stages of execution and may compile the code using different models. We should also remember that the rest of the Java subsystems still run, such as the garbage collector, and threading kernel.

While JIT does bring big improvements in speed, it unfortunately also carries two main risks: code bloat and compile stalls.

First, depending on the approach taken, at some time the executing block of Java byte code will need to be compiled into native processor instructions. This obviously requires processing power. (Improvements will be possible via pre-JIT to reduce these risks; however, Ahead-of-Time disadvantages may be introduced.)

Second, once the code has been compiled, it needs to be stored. On a mobile device this may mean Flash ROM or even RAM. As Java byte code is abstract, it can represent more functionality in less space than its native processor equivalent instructions – especially in the case of a RISC processor. This code bloat can be anything – up to a factor of 10. Another problem with JIT is the fact that code will be compiled into native instructions, even if that block of code is seldom executed.

Speed	There will be initial compile stalls as the Java byte code is compiled. However, once compiled there will be a big and noticeable improvement in performance. It may be possible to improve performance even more depending on how the JIT compiles the code. For example, the byte code and native code may be optimized
Memory Requirements	More storage will be required to store the native compiled static footprint. Also, the complexity and size of the Java VM will increase as we are now carrying a compiler as well as the Java subsystems and our applications.
Power Consumption	Extra processing power is required to do the compilation.
Licensing and Silicon Cost	Cost of licensing the software VM.

TABLE 3 JIT

Again, depending on the device, processor, and implementation, this effect may not always be noticed by the end user, depending how well it's been hidden (see Table 3).

Ahead-of-Time Compilation

A good way of speeding up a JVM is to use ahead-of-time compilation. This has the added benefit of reducing compile stalls on the device, and can be performed for the bottlenecks to control code bloat. However, the main disadvantage of this is that separate binaries will have to be maintained for different platforms – the platform-independent advantages are lost. Alternatively, only the device's "core" applications can be compiled ahead-of-time, while any dynamically downloaded programs could run interpreted.

As with JIT technology, a common misconception with ahead-of-time is that the application will run as quickly as an equivalent native C application. This isn't the case as the application is still a Java application and, once again, still runs within the Java domain (see Table 4). The rest of the Java subsystems still run, such as the garbage collector and threading kernel,

Speed	Big performance over interpretation for those parts that are compiled. Any downloaded applications that aren't compiled may suffer much poorer performance.
Memory Requirements	Similar to pure interpreter.
Power Consumption	
Licensing and Silicon Cost	Cost of licensing the software VM and compiler.

TABLE 4 Ahead-of-Time Compilation

Smart JIT

To reduce the code bloat risk associated with regular JIT, Smart JIT can be used (see Table 5), which compiles only the parts of the program that prove to be the bottlenecks. Furthermore it may limit the amount of complied code stored by throwing away old native code in order to create space for newly compiled blocks. This method can bring some limited code bloat, compile stalls, and profiling stalls – depending on the approach taken. Whatever approach is adopted, the extra effort is paid for in one way or another.

Speed	Big performance improvement over interpretation. Still may have some compile stalls. However, these are limited as the JIT is now "smarter" and only focuses on bottleneck code. Some initial profiling needs to be done that obviously takes up process ing power.
Memory Requirements	Some extra RAM is required for the code bloat parts of the application. Also the VM is now more complex, so static space is probably required to account for this.
Power Consumption	Depending on the approach, processing power will be required to do the compilation and the smart profiling work.
Licensing and Silicon Cost	Cost of licensing the JIT.

TABLE 5 Smart JIT

Hardware Accelerators

An accelerator is a hardware solution that typically "bolts on" to the side of an existing heavyweight processor. The accelerator can't execute anything on its own. Rather it can be thought of as a hardware Java adapter that uses a heavyweight processor to speed up the Java execution. It effectively means that the main processor can be used to execute Java byte codes by using the processor's native instructions to microcode the Java byte code.

The use of a hardware accelerator means there is no code bloat as the native instructions are never stored. It also means that there are no software compile stalls, as the Java byte code to native instruction translation is done in the hardware.

The software-based JVM is still required. The accelerator vendor typically modifies this so that it uses the hardware accelerator in place of its main interpreter loop and byte code execution unit.

The following describes how an accelerator can be used on a "two-sided" smartphone, where one side is used for the 3G baseband critical tasks, and the other is used for the PDA-type noncritical applications.

Speed	Very fast. So far, this is the fastest way of executing Java byte code. However, it may drastically reduce the performance of the main processor, as this now takes on the burden of running the JVM and applications.
Memory Requirements	Very low, since native code is never stored.
Power Consumption	Power needed to do the byte code to native translation and run the JVM via a heavyweight processor.
Licensing and Silicon Cost	• JVM needs to be licensed. • If an additional processor is used so that the main processor can be left alone, then this must be licensed. • If an additional heavyweight processor is used so that the main processor can be left alone, then an additional operating system must be licensed. • The accelerator must be licensed.

TABLE 6 Hardware Accelerators

Single Chip Solution

In this configuration the baseband chip that manages the 3G protocol stack is also used to execute the Java subsystems and Java applications (after being translated into native instructions by the accelerator). This means the main chip is used to run the following:

- Baseband tasks + original phone software
- Java subsystems
- Java applications

Dual Chip Solution

It's important to remember that this may be contained on a single piece of silicon.

In order to prevent any degrading in the performance of the main baseband processor (which we shall refer to as the *Master*), we have introduced another processor to take on the work of the PDA (we shall refer to this as the *Slave*). This circumvents any performance, stability, or security problems on the more "critical" side of the phone. The Master and Slave typically communicate via a system bus, such as a VCI compatible bus.

While hardware acceleration is a much improved method of running Java byte code, there are a number of potential issues with this approach:

- Master processor work load increase
- Licensing cost of extra heavyweight native processor
- Silicon cost

Master Processor Work Load Increase

First, if the accelerator is bolted on to the side of your existing processor, then it will have to also carry the burden of a Java Virtual Machine in addition to its normal duties. Typically, the accelerator looks out only for Java byte codes. When it "sees" one, it does a processor context switch into "Java mode," then pumps the native instructions into the regular processor for execution. The regular processor still needs to do the actual "grunt" of the processing.

In addition to this, the native processor also needs to run the Java subsystems, such as garbage collector, threading kernel, dynamic class loader, and verifier. Again this requires processing bandwidth on behalf of the main processor.

Licensing Cost

If the dual-chip solution is used so that the main processor (Master) doesn't have to take on the extra workload, then an additional (Slave) processor will be required – the one to which the accelerator will be bolted (see Table 6). This will obviously incur additional licensing costs, which, in the worst case, may be double.

Silicon Cost

If the dual chip solution is used, then the additional processor will mean that the silicon cost will also increase. Again, at worst, this may be double.

Speed	About as fast as it can get. A core set of the Java byte codes is now true native machine instructions. The remaining higher-level byte codes then use these as microcode. The native Java processor vendors will provide an optimized version of a Sun-compliant J2ME VM built specifically for the native processor. The hardware and firmware are developed and provided in synergy by one company. This means there can be continuous improvement for performance gains for both hardware and firmware parts. The Java subsystems are developed using optimized Java byte codes and run on the Java processor - not on the main processor.
Memory Requirements	About as low as it gets. Byte codes are executed directly on the Java chip, so they are never compiled and stored. Also the Java subsystems are developed using optimized Java byte codes. Therefore, considerably less memory is required to store the subsystems code image.
Power Consumption	About as low as it gets. Typically native Java processors will require about the same number of transistors as a hardware accelerator – without the requirement for a native processor. This means that the JVM doesn't have the additional power over head of a native processor.
Licensing and Silicon Cost	Licensing cost of the native Java processor. (Note there are no operating system or separate JVM licensing costs or extra heavyweight processor licensing costs.)

TABLE 7 Native Java Processors

Native Java Processors

The next logical step for Java is to use native Java processors, where the native instructions are in fact Java byte codes. This has the advantage of executing actual byte codes in a similar fashion to how a native processor executes its instructions, which means that no translation or interpretation of Java byte codes to another machine instruction set is required. Typically, the processor directly executes a core set of byte codes. This, in effect, means there is a core set of byte codes that are true native machine instructions (see Table 7).

Other more complex instructions are implemented by using the core set of byte code machine instructions as microcode, and the few high-level byte codes are executed by firmware, which uses both the microcoded byte codes and the core native byte codes. Any improvements in the core native byte codes have an immediate effect on the higher-level byte codes, which presents a number of interesting optimization opportunities.

Looking back at our example of the two-sided smartphone, we use a Master heavyweight processor to process the critical heavy real-time 3G communication tasks, and a lightweight Slave Java native processor for the PDA side of things.

It's probably a fair statement to say that the 3G mobile communications processing will only become more complex and time critical as capabilities of the devices and networks increase. Therefore, this processor can be left alone to do its intended job of managing mobile communications.

The Slave Java processor is used to provide the processing power for the PDA applications.

As with the other VM options, native Java processors can also be field upgraded. One drawback of a native Java processor is that if any legacy native code needs to be run then this can only be executed on the Master processor as the Java chip can only understand Java code.

Which Way to Go?

A JVM may provide acceptable performance for a powerful desktop system running at a high clock speed with megabytes of RAM. However, when faced with the constraints of an embedded environment, it may well falter when faced with, for example, 1 megabyte of RAM, no virtual memory, no hard drive, and a very low clock speed.

As we can see there are a number of alternatives when considering a JVM for an embedded environment. Still, the decision as to which way to go depends on many factors, including the device, processor, clock speed, available RAM, sophistication of user applications, and the required usability levels.

In this article, we've looked at the software techniques that have become more widely used in embedded environments with successful results, and at hardware alternatives that are designed to provide the next generation in performance while keeping memory and processing power to a minimum.

In terms of hardware-based VMs, accelerators are the next step toward this "next generation." Native Java processors are a generation further on and are regarded as the ultimate goal. This will open up a new set of exciting possibilities for mobile, wireless, and embedded applications.

AUTHOR BIO

Glenn Coates has been a software engineer for nine years. For the last four years he has worked with mobile devices and Java developing products, such as smartphones, microbrowsers and digital set-top boxes. Glenn holds a degree in computer science and is also a Sun-certified architect for Java technologies. He works for Vulcan Machines as a VM architect developing a Java native processor called Moon. See www.vulcanmachines.com

glenn@vulcanmachines.com

Freedom Through Constraints

Designing games for constrained platforms

WRITTEN BY
TOM SLOPER

Today we have new gaming platforms. There are games for wireless phones and PDAs. Some calculators, watches, messaging pagers, and other handheld devices are also capable of supporting games. The purpose of this article is to apply basic game design principles to these devices.

I've worked as a designer and producer of electronic games since 1979. My first electronic game design was for a game watch (Game Time by GCE, circa 1981–82). I've designed handheld game LCDs, board games, dice games, and games for videogame consoles, computers, and the Internet. I'm not a programmer. The ideas in this article are based on general principles of game design – not on details of the technology.

When designing games for a new gaming platform, the first fact you must face is the constraints of the system. These platforms aren't generally intended to support games. These devices can't do what a PC or set-top console can. But that's okay.

In fact, it's more than okay!

With the capabilities of PCs and consoles increasing by leaps and bounds, the game designer has been pressured to make increasingly complex games. But on a constrained system, expectations are lower, so the pressure is off the designer to deliver complexity. You can look at these constraints as limitations that force you into simpler game play. Or you can see these constraints as a license to make simpler games. And in my opinion, the simplest games are usually the best.

Editor's Note: For all of you wrinkling up your nose because this is an article on design, don't be too quick to wreck your rugged good-looks. If you're developing games for constrained devices, then there's a chance you'll be involved to a degree in game design. In fact, I suspect a lot of you will be wearing both design and development hats in your game development efforts.

Tom Sloper was designing games back when Man was still trying to figure out how to roast those prime haunches of mastodon meat... okay, that's an exaggeration, but he was certainly around close to the beginning of things, in terms of the games industry. Where better to learn than at the feet of the master?

–JB

The Specifics

Here are the main areas you have to consider when designing a game for a constrained platform:

- Target audience
- Graphics
- Sound
- User interface
- Memory storage
- RAM capability
- Other special limitations or capabilities of the system
- Target audience
- Who's writing the checks

Let's consider each area in detail.

Target Audience

First, picture your audience naked. Okay, enough of that. Now picture them with clothes on, playing your game. Where are they? Why are they playing your game? Who are they? How old? What gender? What income bracket? What nationality and language? Be realistic, not idealistic, in this consideration.

Constrained gaming devices are typically portable. Your end user might be standing in line at Starbucks. Or maybe sitting at a bus stop (hmm, no – maybe a taxi stand or airport). Think of more examples where your user is while playing your game. "Where" and "why" go hand in hand.

The user of one of these new devices is probably an adult. An adult, most likely male, with money to spend on electronic gadgets. An "early adopter." These are mostly folks who spend a lot of their time multitasking. Not a moment is wasted: every idle moment affords the busy multitasker an opportunity to check text messages or stock prices... or to play a game. So how about this for a wacky idea: a game that requires the player to use those multitasking skills. A game that can be played in one-minute increments, in-between other tasks.

If your research shows that teen girls are increasingly using game-capable pagers, it might make sense to design a girl game for those pagers. But the teen girl demographic has historically been dismal for games. You might surprise everyone and break the historical trend (and garner headlines and big bucks), but is that really the hill you want to die on? Of course, if some company comes to you and offers to pay you to design/develop it, that's another matter.

Anyway, once having thought through who your target audience is, you probably have a few initial game ideas you want to explore. The next step is to consider the technical aspects, after which you may have to rethink your ideas, culling and/or refining them.

Graphics

When working with a constrained platform, get all the information you can about the graphics capability of the system. There are some PDAs that support hundreds of colors, but right now the widest audience will be found for wireless phones, PDAs, and some types of multipurpose pagers, usually with black-and-white displays.

Consider the size and resolution of the display. One of the worst-case scenarios today is a Nokia phone with a black-and-white display capability (or should I say "limitation") of 90x40 pixels. 3G (third-generation) phones will surely be better than that; 4G will likely be even better. But all in due time.

Can your system display animated graphics, or do you have to work with still images only? Is it black-and-white, or can it support color, or shades of gray?

Using the smallest readable font, how many letters can you fit across the display? Does the system come with a font, or do you have to design one? Is your font variable-width or fixed-width? I once worked on a Game Boy game with a Japanese company. They had translated the game's story text into English, which was to be displayed in a text window, two lines by 16 characters wide. The text hadn't been written with this limitation taken into consideration; the user would have to click endlessly to read the lengthy text, sometimes just to read the final word of a sentence in an otherwise empty window. There was no question of enlarging the text window or shrinking the font, so it fell to me to rewrite the text to fit more compactly in the window.

Find out what your system is capable of visually, and make adjustments to your game as needed.

Sound

Maybe your user is a college student, playing a game rather than listening to a boring lecture. Or a guy in a cubicle, trying not to broadcast the fact that he's taking an unsanctioned game break. If so, the user will want to have the ability to disable sound. But it's important to have sound in your game. Every action in the game should be accompanied by a sound.

Find out what the system is capable of. What kind of sounds can it play? Can it play music? Can it play multiple sounds simultaneously? Often it's the case, with a constrained platform, that the answer to the last one is "no." It often occurs that two game sounds need to be played at the same time. So it's necessary for the sounds to be prioritized so that the more important sound is heard.

Let's say your game has a sound to indicate that the player character is walking, and a sound to indicate the death of the player character. If the player character is walking when he gets killed, it's the death sound, not the walking sound, that needs to be heard.

The highest-priority sound in your list is the "game over" sound; you can figure out the rest from there.

User Interface

Consider the methods by which your user can interact with your game. If you've seen other games on your platform, consider using a similar interface scheme. That way, players of other games don't have to learn a whole new paradigm when picking up yours.

Phones have a 12-button dial and a couple of other buttons, not all of which are available to you for game play. There's no joystick or mouse. You'll need to check which buttons can be used for the game. What will happen if somebody calls while the user is playing? How can the player get out of the game to make or take a call? Typing text is a pain on a phone; try not to make the user do stuff that isn't fun.

Palm-style PDAs have a stylus, and a couple of other buttons. The stylus should be the main input mechanism – probably the sole input mechanism – for the game.

Other PDAs, and some pagers (like the Motorola PageWriter) have QWERTY keyboards. Keyboards are both good and bad when it comes to gaming. The good thing is there's already a game interface "language" that's been established that you can use. The bad thing is that too many game input buttons can make things confusing for the player. Limit the use of the keyboard to the most logical and intuitive keys, and let the player configure them (and make it easy to find the config menu).

Consider how the user can exit the game, go do other tasks, and come back. Consider how the user can quit a game that's going badly and start over. Your game must be intuitive; your user won't be carrying an instruction booklet everywhere. There shouldn't be a need for an instruction booklet at all.

Data Storage

As a designer, I don't necessarily need to know exactly how many bytes of data the platform can hold, but to know whether it's "very little" or "fairly roomy" helps me put things in perspective. Can your game system support the kinds of features you initially envision for your game? For example, can it store the graphics for multiple screens? Can it store animations and sounds and game logic?

RAM

Not being a programmer, it took me a long time to realize the difference between data storage and RAM, and the impact it has on a game. For the nontechnical designer, RAM size impacts how much stuff can go on during a particular level or stage of your game. Your game doesn't need to load into RAM any sounds not used in the current level, for instance. Once again, get an idea of how much RAM is available, and what impact it will have on your initial basic idea.

Other Limitations

All your dreams for your game idea will probably be dashed once you work out what

Hello World

Meet some of the country's best programmers.

- Competing online each week for cash prizes.
- Coding against the clock and each other in real time.
- Qualifying for a chance to win $100,000 in the TopCoder Invitational.

Join us at www.topcoder.com or stop by our exhibition booth (#100) at JDJ Edge.

TopCoder Programming Competitions:
Profit from Performance.

[TOPCODER]

Performance. Credibility.

you can't do on the intended platform. In this case, just close your eyes... imagine the world from the other side of the mirror... and figure out what you can do.

Look for inspiration in new games like EA's Majestic. Find new play paradigms that take advantage of what the hardware was designed for. Calculators are good for math: think about math games and puzzles. Wireless devices that have a QWERTY keyboard would be good for text-based games, involving sending and receiving messages. Palm-type PDAs let you write by hand and touch any part of the display with a stylus. This can be better even than a joystick or mouse for solitaire gaming.

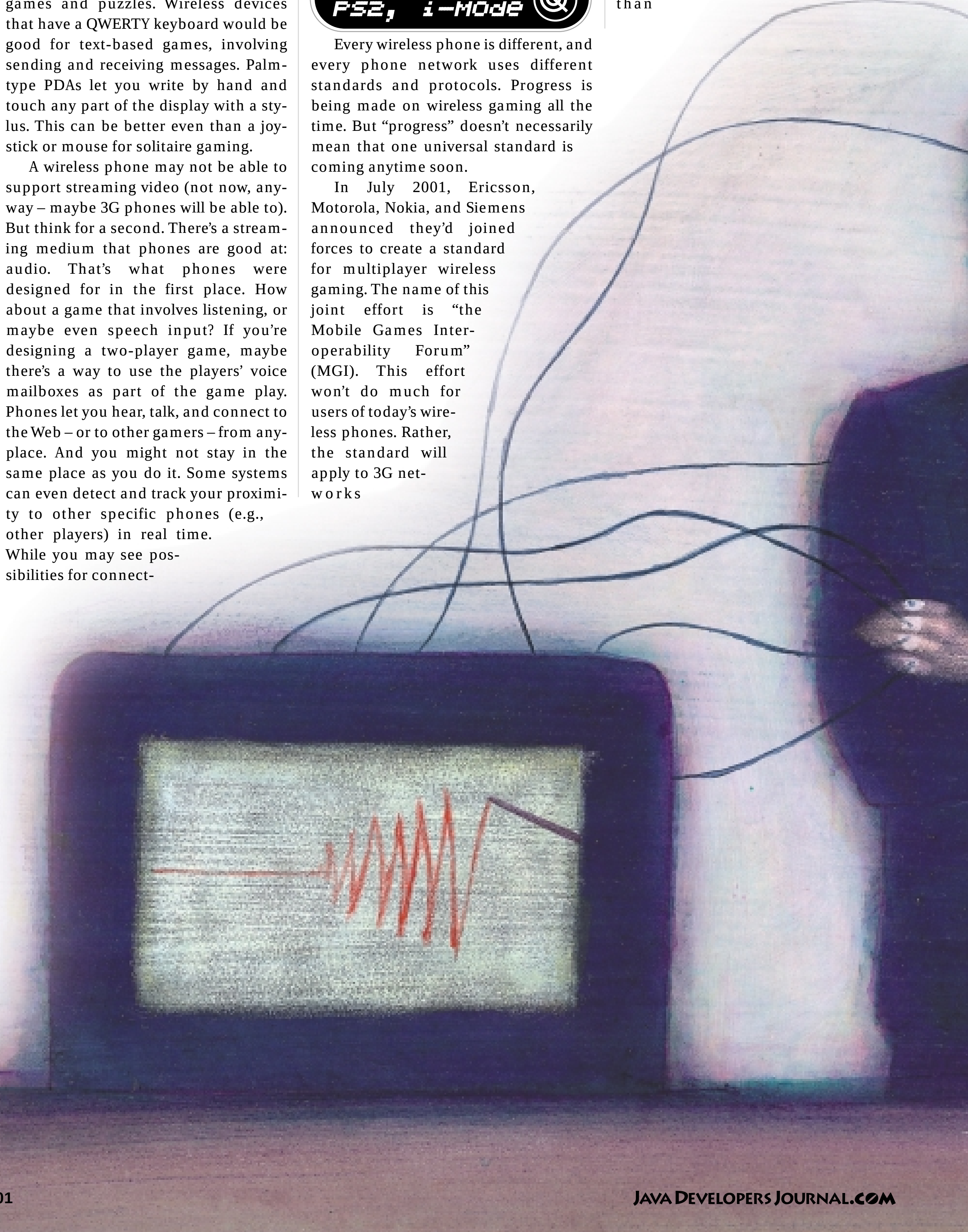
A wireless phone may not be able to support streaming video (not now, anyway – maybe 3G phones will be able to). But think for a second. There's a streaming medium that phones are good at: audio. That's what phones were designed for in the first place. How about a game that involves listening, or maybe even speech input? If you're designing a two-player game, maybe there's a way to use the players' voice mailboxes as part of the game play. Phones let you hear, talk, and connect to the Web – or to other gamers – from anyplace. And you might not stay in the same place as you do it. Some systems can even detect and track your proximity to other specific phones (e.g., other players) in real time. While you may see possibilities for connecting multiple gamers through wireless phones, for now perhaps you should limit your thinking to two players max. The world isn't yet ready for Massively Multiplayer Wireless Phone Gaming. Well, even if it is, the technology isn't quite there yet.

MGI, 3G, PS2, i-Mode

Every wireless phone is different, and every phone network uses different standards and protocols. Progress is being made on wireless gaming all the time. But "progress" doesn't necessarily mean that one universal standard is coming anytime soon.

In July 2001, Ericsson, Motorola, Nokia, and Siemens announced they'd joined forces to create a standard for multiplayer wireless gaming. The name of this joint effort is "the Mobile Games Interoperability Forum" (MGI). This effort won't do much for users of today's wireless phones. Rather, the standard will apply to 3G networks and phones. Don't expect to see developer kits on this until 2002.

In addition, NTT DoCoMo is involved in an effort with Sony and six other wireless companies to develop a wireless gaming service based on Sony's PlayStation 2 system (*Hint:* It's sure to be better than



Author Bio
Tom Sloper produced and designed games at Activision for 12 years, and is now consulting and designing independently.

WAP; no hints yet as to how it'll compare to 3G). NTT DoCoMo and Sega are also working on games that will link i-mode phones to Sega arcade games.

Whatever the system, get the specs, and you can design a game for it.

Target Audience

This isn't a mistake: I intentionally listed "target audience" twice. If you're wearing a hardhat, please take it off for a moment while I hammer this in: you must consider the target audience first.

It's unlikely that any grandmas will be playing your PDA game. And it's unlikely that many kids will be playing your game, since we're talking about hardware used by adults. The main users of the devices under discussion are early adopters. Early adopters might enjoy games involving money, or cutting-edge technologies, or time management.

And finally, you also need to think about...

Who's Writing the Checks?

The typical game designer (at least, for the constrained platforms we're discussing here) is not necessarily going to be selling the game directly to the end user. Wireless games are billed and tracked by phone services, not by game companies or game programmers. What's the business model in your case? You need to know.

Where's the money coming from for your project? Perhaps a PDA manufacturer hires you to design a game that takes advantage of the features of a new PDA they want to tout. Perhaps your assignment is to make an advergame, a regular game that contains product placements or billboards for a company interested in selling its products to the audience playing your game.

Your client is the company that's paying you to produce this game. That company, not the end user, is your direct customer. Remember the old adage, "The customer is always right." If you want to make money designing and/or developing games, you need to listen to the client's requirements and desires, and deliver accordingly.

Conclusion

First you start with the known parameters: the capabilities of the system, the preferences of the target audience, and the thinking behind the hand that writes the check.

Don't try to make the platform do things it wasn't designed to do. Rather, try to find ways to have fun with those special capabilities it *was* designed to do.

Because the constraints of these devices don't allow you to make a Myst or a Quake, you are freed to go back to the classic purity of small, simple games.

To discuss this article go to www.sloperama.com/advice/bulletinbd.htm.

tomster@sloperama.com

Meet TopCoder

A new kind of programming community.

- Showcase your talents in the great American tradition of competition.
- Compete for cash, ratings and bragging rights.
- 256 of the best will compete for a chance to win part of the $250,000 purse at the 2001 TopCoder Invitational.

Join TopCoder and get a piece of the action.

TopCoder Programming Competitions:

Profit from Performance.

[TOPCODER]

Performance. Credibility.

A Beginner's Guide to Writing Applications for the MID Profile

J2ME and the enterprise

Part 3 of 3

WRITTEN BY
JASON BRIGGS

In Parts 1 and 2 of this series, we covered the basic features of the various MIDP APIs. We looked at creating and packaging a MIDlet, creating a user interface, and some basic graphics operations. We also discovered how to store data with the record management system and how to communicate over the network.

This time around, we're going to talk about MIDP in the enterprise sense (which has nothing to do with "Star Trek"), and put together a basic example that shows how a MIDlet fits into the big picture.

As mentioned before, and it bears repeating, a J2ME developer will undoubtedly not be spending all of his or her time developing on the client side. J2ME applications, particularly MIDlets, will involve a degree of interaction with the server – and unless you're working on an exceptionally large system with a very big team, it seems likely you'll be the developer with "fingers in many pies."

What does that mean exactly? To put it in the simplest terms possible: if you've been neglecting your J2EE and J2SE education in favor of Micro Edition, now's the time to dig out those dusty textbooks and old copies of ***JDJ*** and do some serious reading.

Client to Server Communications

In standard Java, there are a number of ways a client-side application can communicate with another machine. We have everything from raw socket connections where you have to pretty much write your own communications protocols yourself, up to Remote Method Invocation (RMI) from one VM to another, on the same or a different host. On MIDP, however, we're reduced to the basics of network communications (for an introduction to networking in MIDP, see my previous article, in ***JDJ*** (Vol. 6, issue 8) – where HTTP will probably be the most common methodology.

Typically we'll see the following configurations for basic enterprise MIDP applications: a servlet talking directly to the database (as shown in Figure 1)or a servlet talking to an application server that talks to the database (or in some cases, with no database at all) as seen in Figure 2.

Configurations aside, there should be a layer between the MIDlet and the enterprise data, however it's held, to preserve a level of abstraction. For example, an HTML-based interface may view hundreds of rows of data at any one time, whereas a MIDlet may want to view only 10 or 20. Assuming that the enterprise layer has already been written, there may be no real point in adding yet another method to an EJB, just to retrieve 10 rows at a time from the database for a new MIDlet application – rather than retrieving the entire data set. Instead, write a servlet "interface" to the EJB that loads the data into the client's session and then the MIDlet may call the servlet any number of times to get subsets of required data.

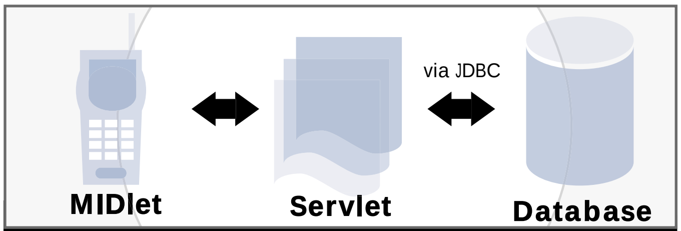


FIGURE 1 Basic configuration with servlet talking directly to database

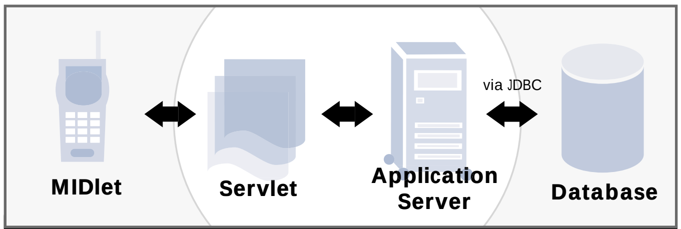


FIGURE 2 Basic configuration with application server

The Example So Far

As it stands, last month's example MIDlet – the PhoneBook application – is fairly simple. You can list the contents of the phone book and add or remove contacts. If you remove the MIDlet suite from the phone, however, the contents of your phone book are lost.

To fix this problem, we need a repository – somewhere to put the data. One possibility here is to store the data in a directory server and access it directly in a servlet, which can expose simple methods for the MIDlet (or any other interface) to access the data. Another option is a database, with an EJB to handle data access. Again a servlet becomes a middleman between MIDlet and EJB/Database.

Rather than go through the rigmarole of setting up a database, the EJB used in this month's example writes its state out to a text file, and I've used the Java 2 Enterprise Edition SDK as the EJB container. Please note, to use the EJB, you'll have to set File-Write permissions in the policy file for the AppServer (server.policy, located in the j2sdkee/lib/security directory), which is a quick and dirty cludge I've used in this case, but wouldn't recommend in

Next Month

A Storm in a Coffee Cup
Embedded real-time Java components for hardware interfacing
by Mike Frerking and David Hardin

Making Java Work in Embedded Devices
Leverage business from a new perspective
Part 2
by Vincent Perrier and Steven Schwarz

Core J2EE Patterns
by John Crupi

J2EE Panel Discussion
by Alan Williamson

Journeyman
Making the Move to J2EE
by Charles Arehart

JAVA DEVELOPER'S JOURNAL

an actual system.

Note: Look for default permissions in the policy file and change the line:

```
permission java.io.FilePermission
"*", "read";
```

to

```
permission java.io.FilePermission
"*", "read,write";
```

Listing 1 shows the "guts" of the PhoneEJB (a stateful session bean). There are five implemented methods: ejbCreate, ejbRemove, add, remove and getPhoneBook.

- ***ejbCreate:*** Called when the EJB is created, and takes one parameter – user. It loads names and phone numbers from a data file of the same name (if it exists), and stores the information in a map.
- ***ejbRemove:*** Called at the end of the EJB life cycle, at which point the bean can be garbage collected. In this case, the method writes the data out into a data file using the user details.
  - ***Add:*** Takes a name and phone number as string parameters and saves these into the map.
  - ***Remove:*** Takes a name as a string parameter and removes the matching record from the map.
  - ***getPhoneBook:*** Returns the map to the caller.

Nothing too complicated here, of course. The next step is to provide an easy way for the MIDlet to call this EJB. Listing 2 shows part of the doGet method in TestPhoneServlet.java.

Lines 1–6 instantiate an instance of the Phone EJB.

Line 8 retrieves the "action" parameter from the servlet request.

If no action has been specified, we assume that the caller wants to retrieve the data, so lines 9–21 call the getPhoneBook method on the EJB, to return the map and then iterate through the keys in the map, writing the name and phone number to the servlet output stream, with one record per line.

If "add" has been specified as the action, lines 22–32 get the name and phone number parameters from the servlet request, check to make sure they're not null, and then call the add method on the EJB.

If "remove" has been specified as the action, lines 33–42 get the name parameter from the request, check that it's not null, and then call the remove method on the EJB to remove the associated record.

In both cases, the servlet writes out an empty line if it succeeds, or an error message beginning with "%ERROR" if an unrecognized action is sent.

(*Note:* For my testing purposes I've used Apache's Tomcat servlet container.)

MIDlet Revisited

We now need to add functionality to the PhoneBook MIDlet to access the servlet. Listing 3 shows the first of the new functionality – the initRecordStore method. This method calls the servlet using the URL:

```
http://localhost:8080/servlet/Test
PhoneServlet
```

You'll recall from the servlet code, if you pass no parameters in this manner, then a list of records is returned by the servlet. If no errors occurred (a return of "%ERROR" from the servlet indicates something went awry), then the current contents of the record store are deleted and replaced with this list.

To add each record to the store, the initRecordStore method calls save (implemented in last month's installment). Note that the save method has also been modified to take a Boolean parameter (sendToServer), which indicates whether the data should be saved to the server in addition to the record store. As we're initializing the record store, we don't want to save it to the server.

Listing 4 shows the changes to the save and erase methods. The main difference in these is that the send-to-server functionality has been implemented to ensure that any changes made at the client level are mirrored on the server.

Missing from This Version

Apart from a few more supporting methods and some minor changes, the MIDlet is essentially the same. Of course, the example is hardly what you would call an "enterprise application." It does demonstrate that adding this sort of capability to a MIDP application is fairly straightforward. If we were to extend this MIDlet into part of an enterprise Personal Information Manager suite (for example), then we might want to add more validation and include more data in the phone book (mobile number, fax number, e-mail address, etc).

One "buglet" in the current app is that duplicates are allowable in the client, but in the EJB they'll be over-web services EDGE conference & expo™

October 24–25, 2001

XML EDGE conference & expo 2001

October 22–25, 2001™

Santa Clara Convention Center Santa Clara, CA

FREE Registration

Register for one...attend both

See them First at XMLEdge2001

Fundamentally Improving the Speed, Cost & Flexibility of Business Applications

Many New XML Products Will Hit the Market in Q3 of 2001

Opening Keynoter

Charles Goldfarb

The Father of XML

SYS-CON MEDIA

Owned & produced by SYS-CON EVENTS

WEB SERVICES CEO PANEL

Litwack
CEO
SilverStream

O'Connor
President
Sonic Software

Kutay
CEO
AltoWeb

O'Toole
Chairman
Cape Clear

Slama
CEO
Shinka Technologies

Morris
CEO
IONA Technologies

www.SYS-CON.com

Platinum Sponsor: ca™ Gold Sponsor: ADOS™ Silver Sponsor: SilverStream™ Corporate Sponsors: xml.org™ Intraspect™ Association Sponsors: OASIS™ XML.ORG™

Media Sponsors: XML JOURNAL™ JAVA DEVELOPER'S JOURNAL™ WebServices JOURNAL™ WebLogic DEVELOPER'S JOURNAL™ WebSphere DEVELOPER'S JOURNAL™ COLDFUSION Developer's Journal™ wireless™ SD Times™ Programmer's Paradise

written (since the data is stored in a map and keyed on name, the phone number is overwritten in the case of a duplicate). Either we could store the information differently in the EJB, or disallow duplicates in the MIDlet.

Where to Go from Here?

Even if you're a seasoned embedded/mobile systems developer, J2ME (or in this case, MIDP) introduces new variables into what may have been, up until now, a fairly straightforward equation. Java's strong networking support means your enterprise application may be split across multiple platforms (client and multiple server tiers), in a way it might not have been with traditional development methods and languages.

Parts 1–3 introduced ways you might go about using the various MIDP packages. Of course, this doesn't help when you're actually trying to decide how your application should be distributed across your architecture. In the end, there's no substitute for good old hands-on experience – trying out small projects for yourself. Good luck, and let the MIDP production line roll!

Listing 1 PhoneEJB.java

```java
import java.io.*;
import java.g.*;
import javax.ejb.*;

public class PhoneEJB implements SessionBean {

  String user;
  Map phoneBook;

  public void ejbCreate(String user) throws CreateException {

    DataInputStream dis = null;

    try {
      if (user == null) {
        throw new CreateException("Null user not allowed.");
      }
      else {
        this.user = user;
      }

      phoneBook = new HashMap();

      File f = new File(user + ".dat");
      if (f.exists() && f.canRead()) {
        dis = new DataInputStream(new FileInputStream(f));

        while (dis.available() > 0) {
          String name = dis.readUTF();
          String phone = dis.readUTF();

          phoneBook.put(name,phone);
        }
      }
    }
    catch (IOException e) {
      throw new CreateException("unable to read data file : " +
e.getMessage());
    }
    finally {
      if (dis != null) {
        try { dis.close(); } catch (Exception e) { }
        dis = null;
      }
    }

  }

  public void ejbRemove() {
    DataOutputStream dos = null;
    try {
      dos = new DataOutputStream(new FileOutputStream(user + ".dat"));

      Iterator iter = phoneBook.keySet().iterator();
      String name, phone;
      while (iter.hasNext()) {
        name = (String)iter.next();
        phone = (String)phoneBook.get(name);

        dos.writeUTF(name);
        dos.writeUTF(phone);
      }

    }
    catch (IOException e) {
      e.printStackTrace();
    }
    finally {
```

PREMIERE ISSUE

WebServices JOURNAL

www.wsj2.com

SYS-CON Events presents Web Services EDGE

From the Editor by Jack Martin pg. 5

Guest Editorial by Ajit Sagar pg. 7

Case Study by Robert Diamond pg. 70

Success Stories by Brady Flowers pg. 82

Tips & Techniques by Sean Rhody pg. 94

WSJ News pg. 98

Interview by Robert Diamond pg. 104

IMHO by Jason Westra pg. 110

RETAILERS PLEASE DISPLAY UNTIL AUGUST 30, 2001

Feature: Is It Really XML's Killer App? Real-world Web Services are here to stay — Brady Flowers 22

Security: Authentication, Authorization and Auditing Support for any kind of e-business — Jack Martin 30

Wireless: Enable Your Web Services with WAP Finding common ground — Sean Rhody 38

Tutorial: How to Develop and Market Your Web Services A means for generating more traffic — Ajit Sagar 46

WSDL: Definitions and Network Endpoints Open standards-based technology makes the difference — Israel Hilerio 54

Feature: Deploying EJB in a Web Services Environment Run on any platform — Robert Diamond 62

Web Services and Swing: An Introduction High-performance translations — Jason Westra

Feature: Implementing SOAP-Compliant Applications Exploring technology

SYS-CON MEDIA

```
      if (dos != null) {
        try { dos.close(); } catch (Exception e) { }
        dos = null;
      }
    }

  }


  public void add(String name, String phone) {
    phoneBook.put(name, phone);
  }

  public void remove(String name) {
    if (phoneBook.containsKey(name)) {
      phoneBook.remove(name);
    }
  }

  public Map getPhoneBook() {
    return phoneBook;
  }

  public PhoneEJB() {}
  public void ejbActivate() {}
  public void ejbPassivate() {}
  public void setSessionContext(SessionContext sc) {}

}
```

Listing 2 : TestPhoneServlet.java (part of doGet method)

```
    Context initial = new InitialContext();
    Object objref   = initial.lookup("MyPhoneEJB");

    PhoneHome home  =
(PhoneHome)PortableRemoteObject.narrow(objref, PhoneHome.class);

    phoneejb  = home.create("test");

    action    = request.getParameter("action");
    if (action == null || action.equals("")) {

      Map phoneBook   = phoneejb.getPhoneBook();

      Iterator iter = phoneBook.keySet().iterator();
      while (iter.hasNext()) {
        String name = (String)iter.next();
        String phone = (String)phoneBook.get(name);

        out.println(name + phone);
      }

    }
    else if (action.equals("add")) {
      String name = request.getParameter("name");
      String phone = request.getParameter("phone");
      if (name == null || phone == null) {
        throw new Exception("missing name and/or phone");
      }

      phoneejb.add(name,phone);

      out.println("");
    }
    else if (action.equals("remove")) {
      String name = request.getParameter("name");
      if (name == null) {
        throw new Exception("missing name");
      }

      phoneejb.remove(name);

      out.println("");
    }
    else {
      out.println("%ERROR : unrecognized command");
    }
```

Listing 3 : PhoneBook.java (initRecordStore method)

```
private final void initRecordStore() throws Exception {
    store = RecordStore.openRecordStore("PhoneBook",true);

    String s =
sendToServer("http://localhost:8080/servlet/TestPhoneServlet");

    if (s != null && !s.equals("") && !s.startsWith("%ERROR")) {
      recordEnum = store.enumerateRecords(null, pbcomp, false);
      while (recordEnum.hasNextElement()) {
        int id = recordEnum.nextRecordId();

        store.deleteRecord(id);
      }

      String line = "";
      for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) == '\n') {
          if (line.length() > 0) {
            save(line.substring(0,MAX_NAME_LENGTH),line.sub-
string(MAX_NAME_LENGTH), false);
          }

          line = "";
        }
        else {
          line += s.charAt(i);
        }
      }
      if (line.length() > 0) {

save(line.substring(0,MAX_NAME_LENGTH),line.substring(MAX_NAME_LENG
TH), false);
      }
    }

    recordEnum = store.enumerateRecords(null, pbcomp, false);
  }
```

Listing 4 : PhoneBook.java (save and erase methods)

```
/**
  * save a name and phone number to the record store
  */
  private final void save(String name, String phone, boolean
sendToServer) throws Exception {
    if (isEmpty(name)) {
      throw new Exception("No name entered");
    }
    else if (isEmpty(phone)) {
      throw new Exception("No phone entered");
    }
    else if (name.length() > MAX_NAME_LENGTH) {
      throw new Exception("Name too long, max=" +
MAX_NAME_LENGTH);
    }
    else {
      byte n[] = rpad(name,10,' ').getBytes();
      byte p[] = rpad(phone,0,' ').getBytes();
      byte rec[] = new byte[n.length + p.length];
      System.arraycopy(n,0,rec,0,n.length);
      System.arraycopy(p,0,rec,n.length,p.length);

      if (sendToServer) {
        String send =
"http://localhost:8080/servlet/TestPhoneServlet?action=add&name=" +
rpad(replace(name,' ','+'),10,'+') + "&phone=" + phone;
        String rtn = sendToServer(send);

        if (rtn != null && !rtn.trim().equals("")) {
          throw new Exception("Server error : "+ rtn);
        }
      }

      store.addRecord(rec,0,rec.length);


      n = null;
      p = null;
      rec = null;
    }
  }

 /**
  * erase the current record
  */
  private final void erase(boolean sendToServer) throws Exception
{
    if (currentRecordID < 0) {
      return;
    }

    try {
      if (sendToServer) {
        String send =
"http://localhost:8080/servlet/TestPhoneServlet?action=remove&name=
" + rpad(replace(list.getName(),' ','+'),10,'+');
        String rtn = sendToServer(send);

        if (rtn != null && !rtn.trim().equals("")) {
          throw new Exception("Server error : "+ rtn);
        }
      }


      store.deleteRecord(currentRecordID);
      if (store.getNumRecords() < 1) {
        currentRecordID = -1;
        set(null);
      }
    }
    catch (InvalidRecordIDException irie) {
      throw new Exception("Invalid record");
    }
    catch (Exception e) {
      throw new Exception("Unhandled.");
    }

  }
```

Download the Code!
www.JavaDevelopersJournal.com

JDJEDGE conference & expo

REGISTER TODAY!

GET UP TO SPEED ON WIRELESS APPLICATIONS USING J2ME TECHNOLOGY

J2ME FOR WIRELESS APPLICATIONS

September 23-26, 2001

New York Hilton
New York, NY

The Depth and Breadth of Education to Advance Your Professional Skills

The JDJEdge 2001 Conference & Expo provides your best opportunity to understand how Java technologies can solve enterprise challenges.

Four information-packed JDJEdge Tracks...

Track 1 J2ME – Micro Edition & Wireless
Cutting-edge sessions for software engineers and hardware specialists working on wireless solutions.

Track 2 J2SE – Standard Edition
General Java programming for corporate programmers developing full Java applications, including several introductory sessions.

Track 3 J2EE – Enterprise Edition
Advanced sessions for software architects, Web programmers, corporate developers and consultants developing server-based applications.

Track 4 Working with *I*-Technology
Technical and management sessions for business analysts, corporate systems managers, architects, project managers and CIOs .

Who Should Attend

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

The only event backed by SYS-CON Media and *Java™ Developer's Journal*

JDJEDGE 2001 Features

- Unmatched Keynoters and Faculty
- Over 150 Intensive Sessions and Tutorials
- The Largest Java Expo on the East Coast
- BEA WebLogic™ FastTrack to Certification
- IBM WebSphere™ FastTrack to Certification
- Macromedia ColdFusion™ FastTrack to Certification
- Sun Java™ FastTrack to Certification
- Oracle 9*i* FastTrack to Certification

J2ME

ME 1
Where J2ME Is Headed
Alan Williamson, *Editor-in-Chief*
JAVA DEVELOPER'S JOURNAL

ME 2
Deploying and Managing Java over Wireless Connections
Gerald Wluka, *Business Strategis*
APPSTREAM

ME 3
J2ME in a Wireless World
Kumanan Yogaratnam, *CTO*
ESPIAL

ME 4
Java Technologies for Mobile Devices and Services
Martyn Mallick, *Wireless Evangelist*
iANYWHERE SOLUTIONS

ME 5
Implications on Wireless Development
Dr. Jeff Capone, *CTO*
ALIGO

ME 6
J2ME MIDP Tutorial
Johnathan Knudsen, *Course Writer*
LEARNING PATTERNS

ME 7
How Do You Verify The Reliability of Your Middleware Components?
Jay Weiser, *Manager*
SEAGUE SOFTWARE

ME 8
Java on Mobile Phones
Jay Batson, *CEO*
PINGTEL

ME 9
Writing "Real Java," Not Just "C in Java"
Mike Elliott, *Engineering Applications Programmer*
SUN MICROSYSTEMS, INC.

ME 10
TBA

ME 11
Avoiding Portable Pitfalls
Jason Gola, *Consultant*
CAP GEMINI ERNST & YUNG

ME 12
J2ME Development
Keith Bigelow, *Dir. Product Mgmt.*
LUTRIS

visit...www.sys-con.com/JDJEdge for complete program details

Owned & produced by SYS-CON EVENTS
135 Chestnut Ridge Road,
Montvale, NJ 07645
201 802-3069
Fax: 201 782-9651

EXTREME PERFORMANCE TUNING

Make your Web-based application faster and more scalable

written by James McGovern

There are many articles about basic performance tuning a Java application. They all discuss simple techniques such as using a StringBuffer versus using a String, and the overhead of using the synchronized keyword.

This article doesn't cover any of this. Instead we focus on tips that can help make your Web-based application faster and highly scalable. Some tips are detailed, others brief, but all should be useful. I end with some recommendations that you can present to your manager.

I was inspired to write this article when a co-worker and I were reminiscing about our dot-com days – how we designed for systems that could support thousands of users and had tight code, and how we hit aggressive deadlines. Sometimes there's a trade-off between designing for reuse and designing for performance. Based on my background, performance wins every time. Your business customers understand fast-performing systems even if they don't necessarily understand code reuse. Let's get started on our tips.

How to Use Exceptions

Exceptions degrade performance. A thrown exception first requires the creation of a new object. The constructor in the throwable interface calls a native method named fillInStackTrace(). This method is responsible for walking the stack frame to collect trace information. Then whenever an exception is thrown, it requires the VM to fix the call stack since a new object was created in the middle.

Exceptions should be used for error conditions only, not control flow. I had the opportunity to see code in a site that specializes in marketplaces for wireless content (name intentionally withheld) where the developer could have used a simple comparison to see if an object was null. Instead he or she skipped this check and actually threw NullPointerException.

Don't Initialize Variables Twice

Java by default initializes variables to a known value upon calling the particular class's constructor. All objects are set to null, integers (byte, short, int, long) are set to 0, float and double are set to 0.0, and Booleans are set to false. This is especially important if the class has been extended from another class, as all chain constructors are automatically called when creating an object with the new keyword.

Use Alternatives to the New Keyword

As previously mentioned, by creating an instance of a class using the new keyword, all constructors in the chain are called. If you need to create a new instance of a class, you can use the clone() method of an object that implements the cloneable interface. The clone method doesn't invoke any class constructors.

If you've used design patterns as part of your architecture and use the factory pattern to create objects, the change will be simple. Listed below is the typical implementation of the factory pattern.

```
public static Account getNewAccount() {
    return new Account();
}
```

The refactored code using the clone method may look something like this:

```
private static Account BaseAccount = new Account();
public static Account getNewAccount() {
    return (Account) BaseAccount.clone();
}
```

The above thought process is also useful for the implementation of arrays. If you're not using design patterns within

Essential Technologies for Today's B2B Integration Challenges

Web Services EDGE

PRE-REGISTER ONLINE TODAY **SAVE $300**

SEPTEMBER		OCTOBER	
New York, NY		Santa Clara, CA	
web services EDGE conference & expo EAST COAST	24-25	web services EDGE conference & expo WEST COAST	24-25
JDJ EDGE conference & expo	23-26	XML EDGE conference & expo 2001	22-25

each venue features:

- Over 100 Information-Packed Sessions
- Two Distinct Conferences for Just One Registration Fee
- Pioneers and Visionaries
- Leading Vendors Presenting New and Problem-Solving Products

ColdFusion fast track
SEPT. 24-25, 2001
TWO
CONCENTRATED DAYS OF LEARNING TO JUMPSTART YOUR PROFESSIONAL CREDENTIALS!

Participating Companies, Sponsors and Exhibitors as of June 1, 2001

TogetherSoft, SilverStream, Sitraka Software, Sonic Software, aligo, KENAMEA, shinka technologies, SAVAJE, INFERDATA Corporation, bea, InCert, SEAGULL, DataQuad, idoox, CAPE CLEAR, ParaSoft, MERANT, PointBase, hp invent, pingtel, TopCoder, NOVA LABS, Rogue Wave Software, Bestning, Rational, DataMirror, CodeHound.com, OASIS, ORACLE, Data Junction, ENTEGRITY Solutions, Tower Technology, xml spy, Talking Blocks, WHITEHILL, AdvanTel, XML.ORG, XMLCities, XML Journal, Programmer's Paradise, ColdFusion Developer's Journal, WebSphere Developer's Journal, WebServices Journal, WebLogic Developer's Journal, webtechniques, Java Developer's Journal, SD Times, Wireless Business & Technology

Secure Your Seat Now!

Two Easy Ways to Register

1 On the Web: **www.SYS-CON.com**

2 By Telephone: **201 802-3069**

Be sure to visit www.SYS-CON.com for travel information. Reserve Your Hotel Room Early!

your application, I recommend that you stop reading this article and run (don't walk) to the bookstore and pick up a copy of *Design Patterns* by the Gang of Four.

Make Classes Final Whenever Possible

Classes that are tagged as final can't be extended. There are many examples of this technique in the core Java APIs, such as java.lang.String. Tagging the String class as final prevents developers from creating their own implementation of the length method.

Furthermore, if a class is final, all the methods of the class are also final. The Java compiler may take the opportunity to inline all final methods (this depends upon the compilers implementation). In my testing I've seen performance increase by an average of 50%.

Use Local Variables Whenever Possible

Arguments that are part of the method call and temporary variables that are declared a part of this call are stored on the stack, which is fast. Variables such as static, instance, and new objects are created on the heap, which is slower. Local variables are further optimized depending upon which compiler/VM you're using.

Use Nonblocking I/O

Current versions of the JDK don't provide nonblocking I/O APIs. Many applications attempt to avoid blocking by creating a large number of threads (hopefully used in a pool). As mentioned previously, there's significant overhead in the creation of threads within Java. Typically you may see the thread implementation in applications that need to support concurrent I/O streams such as Web servers, and quote and auction components.

JDK 1.4 introduces a nonblocking I/O library (java.nio). If you must remain on an earlier version of the JDK, there are third-party packages that have added support for nonblocking I/O: www.cs.berkeley.edu/~mdw/proj/java-nbio/download.html.

Stop Being Clever

Many developers code with reuse and flexibility in mind and sometimes introduce additional overhead into their programs. At one time or another they've written code similar to:

```
public void
doSomething(File file) {
    FileInputStream
fileIn = new FileInputStream(file);
    // do something
```

It's good to be flexible, but in this scenario they've created more overhead. The idea behind doSomething is to manipulate an InputStream, not a file, so it should be refactored as follows:

```
public void doSomething(InputStream inputStream){
    // do something
```

Multiplication and Division

Too many of my peers count on Moore's Law, which states that CPU power will double every year. The "McGovern Law" states that the amount of bad code being written by developers triples every year, ruling out any benefit to Moore's Law. Consider the following code:

```
for (val = 0; val < 100000; val +=5) {
    shiftX = val * 8;
    myRaise = val * 2;
}
```

If we were to utilize bit shifting, performance would increase up to six times. Here's the refactored code:

```
for (val = 0; val < 100000; val += 5) {
    shiftX = val << 3;
    myRaise = val << 1;
}
```

Instead of multiplying by 8, we used the equivalent to shift to the left (<<) by 3. Each shift causes a multiplication by factors of 2. The variable myRaise demonstrates this capability. Shifting bits to the right (>>) is the same as dividing by factors of 2. Of course this makes execution speed faster, but may make it difficult for your peers to understand at a later date; therefore it should be commented.

Choosing a VM Based on Its Garbage Collection Implementation

Many people would be surprised that the Java specification doesn't require the implementation of a garbage collector. Imagine the days when we all have infinite memory computers. Anyway, the garbage collector routines are responsible for finding and throwing away (hence garbage) objects that are no longer needed. The garbage collector must determine what objects are no longer referenced by the program and make the heap memory that's consumed by the object free. It's also responsible for running any finalizers on objects being freed.

"Sometimes there's a trade-off between designing for reuse and designing for performance" 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**

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

Track Two: Connectivity
- Smart Cards
- Wireless LANs incl. Bluetooth
- UMTS/3G Networks
- Satellite Broadband

Track Three: Wireless Apps
- Education
- Health Care
- Entertainment
- Transport
- Financial Services
- Supply Chain Management

Track Four: Hardware
- Cell Phones/ WorldPhones
- PDAs
- Headphones/
- Keyboards /
- Peripherals
- Transmitters/
- Base Stations
- Tablets

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

FOR INFORMATION CALL

201 802-3069

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

While garbage collection helps ensure program integrity by intentionally not allowing you to free memory you didn't allocate, this process also incurs overhead as the JVM determines the scheduling of CPU time and when the garbage collector runs. Garbage collectors have two different approaches to performing their job.

Garbage collectors that implement reference counting keep a count for each object on the heap. When an object is created and a reference to it is assigned to a variable, the count is incremented. When the object goes out of scope the reference count is set to zero and the object can be garbage collected. This approach allows for the reference counter to run in small time increments that are relative to the execution of the program. Reference counting doesn't work well in applications in which the parent and child hold references to each other. There's also the overhead of incrementing and decrementing the reference count every time an object gets referenced.

Garbage collectors that implement tracing trace out a list of references starting with the root nodes. Objects found while tracing are marked. After this process is complete, any unmarked objects known to be unreachable can be garbage collected. This may be implemented as a bitmap or by setting flags in the object. This technique is referred to as "Mark and Sweep."

Recommendations for Your Manager

Other approaches can be used to make your Web-based application faster and more scalable. The easiest technology to implement is usually a strategy that supports clustering. With a cluster, a group of servers can work together to transparently provide services. Most application servers allow you to gain clustering support without having to change your application – a big win. Of course you may need to consider additional licensing charges from your application server vendor before taking this approach.

When looking at clustering strategies there will be many additional things to consider. One flaw that's frequently made in architecture is having stateful sessions. If a server/process in the cluster crashes, the cluster will usually fail over the application. For this functionality to happen, the cluster has to constantly replicate the state of the session bean to all members in the cluster. Make sure you also limit the size and amount of objects that are stored in the session, as these will need to be replicated.

Clusters also allow you to scale portions of your Web site in increments. If you need to scale static portions, you can add Web servers. If you need to scale dynamically generated parts, you can add application servers.

After you've put your system in a cluster, the next recommended approach to making your application run faster is choosing a better VM. Look at the Hotspot VM or other VMs that perform optimization on the fly. Along with the VM, it's a good idea to look at a better compiler.

If you've employed several industry techniques plus the ones mentioned here and still can't gain the scalability and high availability you seek, then I recommend a solid tuning strategy. The first step in this strategy is to examine the overall architecture for potential bottlenecks. Usually this is easily recognized in your UML diagrams as single-threaded components or components with many connecting lines attached.

The final step is to conduct a detailed performance assessment of all code. Make sure your management has set aside at least 20% of the total project time for this undertaking; otherwise insufficient time may not only compromise your overall success, but cause you to introduce new defects into the system.

Many organizations are also guilty of not having the proper test beds in place due to cost considerations. Make sure your QA environment mirrors your production environment, and your QA tests take into account testing the application at different loads, including a low load and a fully scaled load based on maximum anticipated concurrent users. Performing tests, sometimes to gauge stability of a system, may require running different scenarios over the course of days, even weeks.

Under no circumstances should you undertake tuning an application without a profiler. We use Optimize it, but Sitraka's JProbe and Numega's profiler are also good. These tools will show you bottlenecks in your code, such as threads that are blocked by other threads, unused objects that survive garbage collection, and excessive object creation. Once you've captured the output of these tools, make simple changes and limit the scope of those changes to things that will make your code faster. Don't worry about reuse, style issues, or anything other than performance. Usually the easily identifiable bottlenecks will be contained within loops and algorithms.

AUTHOR BIO

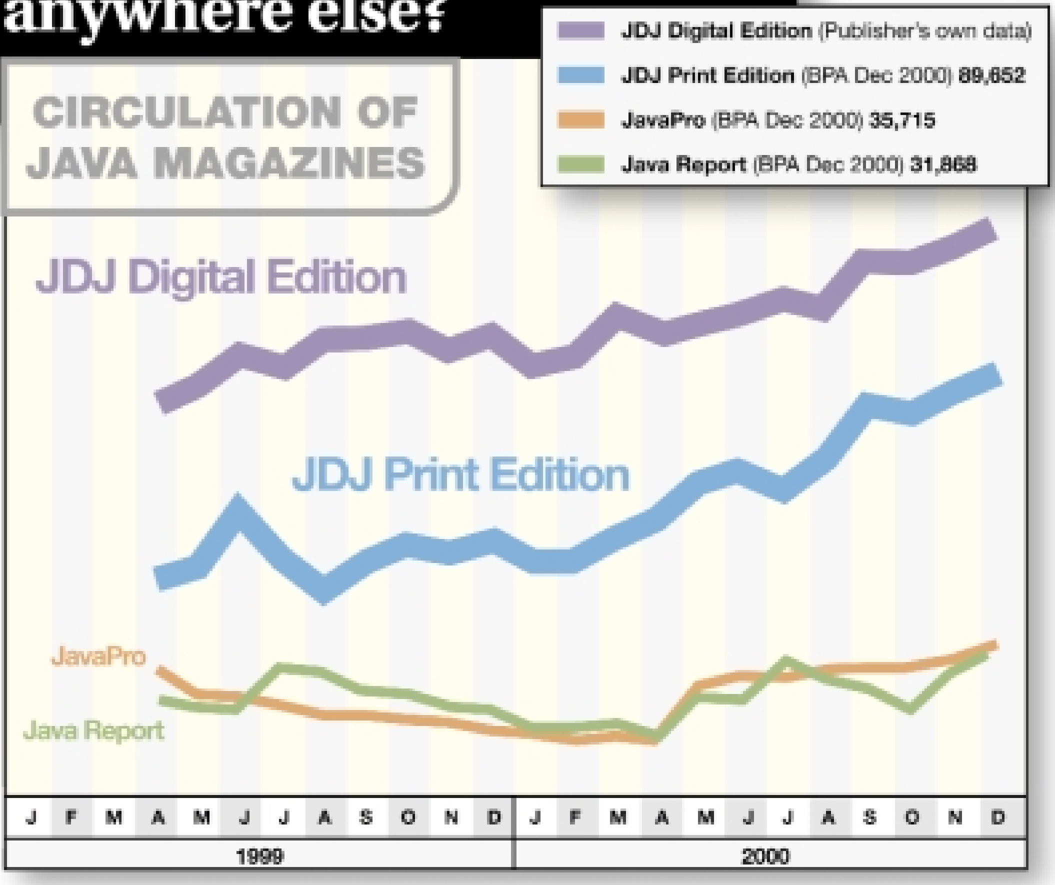
James McGovern is an enterprise architect with Hartford Technology Services Company L.L.C., an information technology consulting and services firm dedicated to helping businesses gain competitive advantage through the use of technology. His focus is on the architecture of single sign-on solutions.

james.mcgovern@htsco.com

"**The easiest technology to implement is usually a strategy that** supports clustering"

It's a no-brainer

Why would you advertise anywhere else?



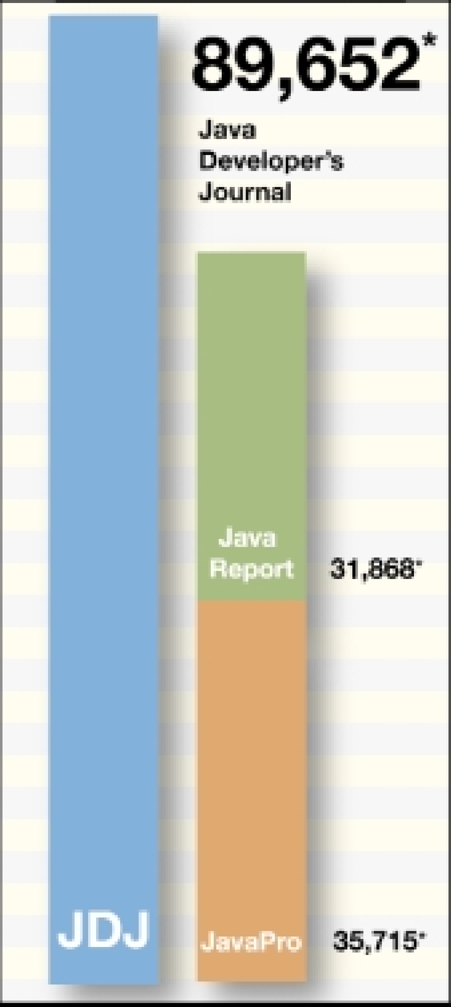
Your ad in *JDJ* reaches twice as many recipients as all other Java pubs combined for 1/2 the cost...*

*independent survey

carmen@sys-con.com

miles@sys-con.com

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

J2ME
J2SE
J2EE
Home

DOT.COM
buyer's guide
java forums
mailing list
• java jobs
• java store

MAGAZINE
advertise
authors
• customer service
• editorial board
subscribe

CONTENT
• archives
• digital edition
editorial
features
• interviews
• product reviews
source code

CONFERENCES
jdj edge
new york, ny
sept 23–26

SEARCH JDJ
Search

What's Online...

September 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 *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, and much more.

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

Subscribe to Our Free Weekly Newsletters

Now you can have the latest industry news delivered to you every week. **SYS-CON** newsletters are the easiest way to keep ahead of the pack. Register for your *free* newsletter today! There's one for Java, XML, Web Services, Wireless, and ColdFusion. Choose one or choose them all.

Subscribe to Our FREE Weekly Newsletters
☐ Java
☐ XML
☐ Web Services
☐ Wireless
☐ ColdFusion
☐ WebSphere
☐ WebLogic
E-Mail Address:
Subscribe

Search Java Jobs

Java Developer's Journal is proud to offer an employment portal for IT professionals. Get direct access to the best companies in the nation. Learn about the "hidden job market" and how you can find it. If you're an IT professional curious about the job market, this is the site to visit.

Simply type in the keyword, job title, and location, and get instant results. You can search by salary, company, or industry.

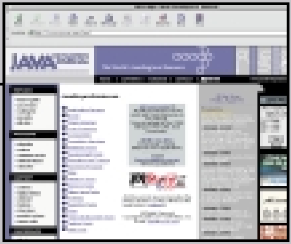
Need more help? Our experts can assist you with retirement planning, putting together a résumé, immigration issues, and more.

JavaDevelopersJournal.com Developer Forums

Join our new Java mailing list community. You and other IT professionals, industry gurus, and ***Java Developer's Journal*** writers can engage in Java discussions, ask technical questions, talk to vendors, find Java jobs, and more. Voice your opinions and assessments on topical issues – or hear what others have to say. Monitor the pulse of the Java industry!

JavaBuyersGuide.com

JavaBuyersGuide.com is your best source anywhere, anytime on the Web for Java-related software and products in more than 20 mission-critical categories, including application servers, books, code, IDEs, modeling tools, and profilers. Check the Buyer's Guide for the latest and best Java products available today.



JDJ SPECIALS

JDJStore.com
$175.99
JDJ, XML-J, CFDJ
ALL THREE!!!
The Complete Works

JDJ BESTSELLERS

1. JProbe Profiler Serverside with 1 Year Gold Support Edition v2.8.1 from Sitraka Windows 2000/98/95 $998.99
2. Chant® SpeechKit™ 3 Professional Edition Speech Recognition Developer's Software $709.99
3. JClass Standard Suite with One Year Gold Support With Source Code from Sitraka $3999.99
4. JFCSuite Enterprise Edition with Source Code v2.0 from Infragistics $2494.99
5. Visual Café Standard Edition v4.0 from WebGain $97.99

Click for a FREE 30-day trial AltoWeb

M-1 Mobile App Server aligo Download Now!

JClass

Get SonicMQ™ PROGRESS SOFTWARE

KADA SYSTEMS

MERANT DataDirect

TopCoder Compete for Cash www.topcoder.com

Sitraka Software

versata

infragistics.com tree · charting calendaring grid & more! Infragistics

JDJ NEWS

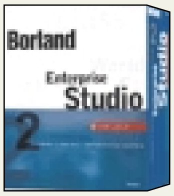
ServletExec 4.0 Released

(*Alpharetta, GA*) – New Atlanta Communications, LLC, announced the availability of ServletExec 4.0, the only commercial product to feature Servlet API 2.3 and JSP 1.2. It also includes the ability to configure JDBC 2.0 data sources, Web application security, resource monitoring, general performance enhancements, and a straightforward installation and configuration process.

The product is available for immediate download at www.servletexec.com.

Borland Announces New Version of Studio for Java

(*Long Beach, CA*) – Borland Software Corporation unveiled the new Borland Enterprise Studio for Java (the Studio), an application development life-cycle solution for building enterprise applications. This new version offers tighter integration between RationalRose, Borland JBuilder, and XML integration.

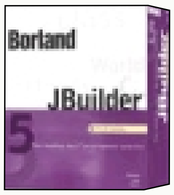


www.borland.com

CocoBase Enterprise O/R Optimized for JBuilder 5

(*San Francisco, CA*) – THOUGHT, Inc.'s, CocoBase Enterprise O/R is now optimized for the Borland JBuilder 5 IDE.

CocoBase enables companies to integrate relational data with EJB applications. Customers using CocoBase with JBuilder can now gain time and labor savings by generating scalable and high-performance CMP and BMP, in addition to JSP with its dynamic mapping capability.



www.thoughtinc.com

Flashline Prepares Corporations for Software Reuse

(*Cleveland, OH*) – Flashline is offering a range of consulting and educational services to teach organizations how to succeed with a reuse initiative. The services address all stages of component-based development (CBD) and reuse adoption.

All services are offered on an individual basis. The readiness review and the reuse analysis survey are included with the purchase of Flashline Component Manager, Enterprise Edition (CMEE).
www.flashline.com

DWL, Sun, iPlanet Announce Transactional CRM Solution

(*New York, NY | Palo Alto, CA*) – DWL Inc., Sun Microsystems, Inc., and iPlanet E-Commerce Solutions, a Sun-Netscape Alliance, announced an EJB architecture-based transactional CRM solution. Powered by iPlanet Application Server and Enterprise Edition software and supporting J2EE, the solution will help financial services and insurance companies leverage and consolidate back-end legacy and administrative systems to create personalized portals that support key CRM functionality.
www.dwl.com
http://sun.com
www.iplanet.com

New Atlanta Acquires JTurbo Business Unit

(*Alpharetta, GA*) – New Atlanta Communications, LLC, has acquired the JTurbo business unit from Ashna Incorporated, a privately held Newark, California–based firm. The acquisition expands the company's software offerings to include Sun-certified Type 4 JDBC drivers for Microsoft SQL Server 6.5, 7, and 2000.
www.newatlanta.com

Sitraka Accelerates Performance Tuning of E-Business Apps

(*Toronto, ON*) – Sitraka announced the full integration and support of Sitraka JProbe with Oracle9*i* Application Server (Oracle9*i*AS). The integration of Sitraka JProbe with Oracle9*i* provides Oracle customers with the ability to identify and eliminate performance issues to ensure the development of high-quality applications that meet their performance criteria.
www.sitraka.com

Segue Upgrades E-Business Reliability Tools

(*Lexington, MA*) – Segue Software, Inc., unveiled the newest versions of two of its e-business reliability products: SilkPilot 2.2, which allows customers to test the interaction between applications, middleware, and servers; and SilkMeter/ASP, the Web-based extension of the SilkMeter licensing technology.

Segue also introduced its "SilkCard" technology, a usage-based debit system that gives software consultants flexibility in deploying Segue's e-business reliability solutions.
www.segue.com

Sybase Unveils Enterprise Portal - Express Edition 2.0

(*Emeryville, CA*) – Sybase, Inc., launched Sybase Enterprise Portal - Express Edition 2.0, which enables enterprises to deploy a personalized, common interface to multiple applications and information sources with single sign-on security. It's a new offering within Sybase's Enterprise Portal product family.

Key features include prebuilt portlets for content feeds and searches; prebuilt support for search, categorization, and personalization; and a full-featured security framework with LDAP and single sign-on support.
www.sybase.com

SilverStream Introduces jBroker Web 1.0

(*Billerica, MA*) – SilverStream Software, Inc., released jBroker Web 1.0, a portable Web services engine and tool designed to build, run, and invoke Web services using Java.

jBroker Web provides a standards-compliant, Web services runtime with a small footprint and flexible architecture.

jBroker Web is free and available for download at http://extend.silverstream.com.

Linux Articles

LinuxBusinessWeek.com is seeking articles for its next few issues. Topics of interest include office applications, security, Linux in the health care industry, embedded Linux, network-centric computing, Linux training, certification, or anything you feel would be of interest to our readers. We also seek case studies, product reviews, book reviews, and how tos/tutorials. E-mail your article ideas to cheryl@sys-con.com.

Follow a Career Path... ...and ask for directions

WRITTEN BY
BILL BALOGLU &
BILLY PALMIERI

Since we started writing this column, we've gotten lots of inquiries from engineers who want to know the best way to reach the level of senior Java server-side engineer.

The trouble that many engineers have in breaking through to this level typically has to do with a lack of key training and the right experience, and, most important, a résumé that doesn't show a clear career path.

Some of this may have to do with a lack of solid career counseling in school. Perhaps aspiring engineers don't seek advice at an early enough stage, or the schools don't know enough about rapidly changing technologies in the real world.

For starters, a typical résumé for a junior or mid-level engineer shows that the person may not have the right degree. An EE, for instance, is less effective than an MS in computer science for a Java server-side engineer.

Next, the person may take the first attractive offer without realizing the consequences.

They may work with languages like Visual Basic, Visual C++, or other MS languages.

After a year or two in their first position, they often move to unrelated technologies by taking a job that seems attractive at the time. They may be working with COM, DCOM, MFC, or Visual C++, which doesn't really give them the right experience. Or perhaps they're working for a large company building internal tools or software with Java but not working on large, scalable, enterprise projects.

With four or more years of what appears to be good, high-tech engineering experience, this person now wants to be a senior Java engineer. But by taking jobs as they come along and picking up whatever technologies are used at each company, this person now has a patchwork résumé that has no cohesion or logical progression to it.

An analogy might be the house contractors who start out building decks, move on to garages, then remodel kitchens. Now they want to be senior lead contractors, building the whole house or even a new development. But what construction company will hire these people to do a job they have no proven experience in?

Although we are talking about smart, talented people here, it's very hard to convince hiring managers that these candidates with the patchwork résumés are the senior Java engineers they're looking for. Even if they get an interview, they won't pass the technical screening.

It's always an uphill struggle.

During the recent high-tech gold rush, companies were settling for less than senior people for these positions. The gold rush is over. Nobody's settling. It's harder now.

So what's a better approach? If you're just starting out, look at your career. Decide what you want to do. Then map out a plan and follow it. What's your ultimate goal? If it's to be a CTO or a "big gun" server-side Java engineer, follow this path:

1. Get a BS or MS in computer science at the best school you can get into. Make sure it's a school that's known for its computer science department, like Berkeley, Stanford, MIT, Purdue, Michigan, or Carnegie Mellon. Managers want to see that you've gotten comprehensive computer science training at a well-known school.
2. Make sure that the program offers you training in distributed computing and object-oriented systems. The school should also offer you a solid foundation in software design and construction for large-scale systems along with OO principles.
3. Remember that it takes several years to reach the senior level. When hiring senior engineers, managers look for eight to 10 years of server-side engineering experience, including C++, Java, and OO technology. With only six or seven years of experience, the person needs to be extremely talented to land a senior position.

Another key to this kind of senior experience is having had full life-cycle ownership of a significant portion of a major project. Many engineers have been individual contributors or worked on a team for a large project, but personal ownership and responsibility through a full cycle is key.

But what if you're in mid-career with substantial commitments and it's not possible for you to go back and start from scratch?

Take as many C++ and Java courses as you can, try to get Java certified, and get your hands on as much OO experience as you can. You may have to take a step back for a while. But unless you're lucky and get a position in a friend's start-up company, it's going to be almost impossible to make the change without some sacrifice.

It's also a good idea to rework your résumé to highlight the Java and OO direction you're heading in and downplay the experiences that aren't relevant to your current goals. Remember, working on many facets doesn't necessarily mean you're a well-rounded engineer.

Would you try to drive across the country without a map? We wouldn't. So why put any less thought or planning into navigating something as important as your career? And when you need directions, ask.

billb@objectfocus.com

billp@objectfocus.com

AUTHOR BIOS
Bill Baloglu is a principal at ObjectFocus (www. ObjectFocus.com), a Java staffing firm in Silicon Valley. Prior to that 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 of ObjectFocus. Before that he 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.

And the Answer Is . . .

WRITTEN BY
BLAIR WYMAN

In the small South Dakota town that is home to my alma mater, there was a section of sidewalk bordering the college campus that had clearly been laid down long before I was born. As I walked to class one fine autumn day, I happened to look down and noticed that one of the squares of this sidewalk had been lovingly crafted to serve as a frame for what appeared to be some meticulously executed handwriting.

The words in the concrete had been strangely "embedded" there with what must have been streams of sand, carefully poured into the still-wet cement. Although I don't know how long the writing had been visible when I first saw it, I imagine that it hadn't always been so. It was almost as though the words had been engineered to "emerge" from the sidewalk, but only after many years of use.

With a medium so exotic, you might expect a commensurately exotic message, but at first blush these words seemed quite mundane. Rather than being the directions to some colossal cache of buried treasure, or a formula for the True Elixir of Youth, the sidewalk carried this simple message:

AS A RULE, A MAN'S A FOOL. WHEN IT'S HOT, HE WANTS IT COOL. WHEN IT'S COOL, HE WANTS IT HOT. ALWAYS WANTING WHAT IS NOT.

Now how could a message so apparently nondescript deserve such a strange and permanent canvas? Could there be some gleaming redeeming kernels of wisdom in this simple rhyme? Or was the artisan just poking fun at his then-distant future by intimating a little piece of undisputed truth?

For my part, I'll try to find some profundity. It's usually there, actually – and even when it isn't you just need to use the word *eyebrows* a few times and no one will notice.

I'll admit that I like my weather "temperate" – not too hot and not too cold. I'll gladly take my stand as a fair-weather friend of fair weather.

Minnesota has two periods of perfect weather each year – one in late spring and one in early autumn – lasting several minutes each. Would it be nice to have this weather indefinitely? I don't think so, but it's purely academic – there's no danger whatsoever of that happening.

However, if life was always easy, we'd never get anything done, right? There's something to be said for the oppressive darkness and bitter cold of midwinter Minnesota. In fact, there are several things to be said about our yearly Arctic blast, and some of them are printable.

One popular assertion around the programming lab is that the quality of a site's code deliverables is directly proportional to the inclemency of its weather. When my family and I moved here many years ago, I quickly observed that we write great code here in Rochester. I also observed that we are subjected to several months of bitter weather each year.

There being no counterexamples in this experiential universe of exactly one data point, I took the logical leap to a committed, unshakable belief in this principle: seasonally inclement weather results in good computer code.

Each of us needs a logically indefensible religious cause to embrace from time to time; we merely carry on the tradition handed down from our ancestors through the centuries. Naturally, my aspirations to the divine could've been aimed at a somewhat higher plane of consciousness, but I've found there is often comfort (and much company) in mediocrity.

People are basically pretty goofy critters when it comes to belief systems, but far be it from me to say which are the goofier and which the goofiest. I mean, as soon as I say, "I've got the Answer!" am I not automatically wrong in at least half the world's eyes, just based on the language I spoke or the cut of my trouser leg?

Perhaps the Answer I should offer up is, simply, "There is no Answer." Do I win the new 6-ton SUV along with that festooned dream home situated squarely in the heart of manicured suburban splendor? Hmmm...eyebrows.

AUTHOR BIO
Blair Wyman is a software engineer working for IBM in Rochester, Minnesota, home of the IBM iSeries.

blair@blairwyman.com

Java Dudes

CAN ANYONE NAME A NEW PROGRAMMING LANGUAGE WITH A NAME LIKE A MUSICAL NOTE?
I KNOW!!! IS IT C-SHARP?
THAT WAS A TRICK QUESTION...
© VladKolarov.com