IS COMPLEXITY **HURTING JAVA?** P.7

# JAVA™ DEVELOPER'S JOURNAL

*The World's Leading Java Resource*

October 2002 Volume:7 Issue:10

JAVADEVELOPERSJOURNAL.COM





RETAILERS PLEASE DISPLAY UNTIL DECEMBER 31, 2002

$5.99US $6.99CAN

0 09281 01751 6

SYS-CON MEDIA

## Building Installers for OS X

NEW RECORD DISTRIBUTION: This issue of JDJ printed 130,000 copies

Make your Java apps available

written by Ian McFarland page 40

JAVA DEVELOPER'S JOURNAL

INTERNATIONAL ADVISORY BOARD

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

EDITORIAL

EDITOR-IN-CHIEF: ALAN WILLIAMSON
EDITORIAL DIRECTOR: JEREMY GEELAN
EXECUTIVE EDITOR: NANCY VALENTINE
J2EE EDITOR: AJIT SAGAR
J2ME EDITOR: JASON R. BRIGGS
J2SE EDITOR: JASON BELL
PRODUCT REVIEW EDITOR: JIM MILBERY
FOUNDING EDITOR: SEAN RHODY

PRODUCTION

VICE PRESIDENT, PRODUCTION AND DESIGN: JIM MORGAN
ASSOCIATE ART DIRECTOR: LOUIS F. CUFFARI
EDITOR: M'LOU PINKHAM
MANAGING EDITOR: CHERYL VAN SISE
ASSOCIATE EDITORS: JAMIE MATUSOW
GAIL SCHULTZ
JEAN CASSIDY
ASSISTANT EDITOR: JENNIFER STILLEY
ONLINE EDITOR: LIN GOETZ
TECHNICAL EDITOR: BAHADIR KARUV, PH.D.

WRITERS IN THIS ISSUE

ASHOK BANERJEE, JASON BELL, JASON R. BRIGGS, S.D. CAMPBELL, DANTE CASTIGLIONE, MATTHEW DODD, GANESH KONDAL, SUNIL KUNISETTY, IAN MCFARLAND, BILL RAY, AJIT SAGAR, MIKHAIL SKORIK, JASON WEISS, STUART WELLS, 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.

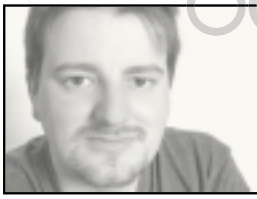
EDITORIAL OFFICES:

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.







ALAN WILLIAMSON EDITOR-IN-CHIEF

# 'Swing Is Swinging Java out of the Desktop'

Over the past few months, I've frequently stated my belief that Java, in its present state, doesn't stand much chance on the desktop against the march of Microsoft's WinForm technology from the .NET Framework. A view, I've discovered, shared by many of you. Before I continue, let me clarify my position so we're all on the same page.

The only frameworks currently available to the Java developer from the standard JDK are, of course, AWT and Swing. Naturally, we're not talking about other client-side technologies, such as SWT and thinlets, to name a few; but more on them later.

The less said about AWT the better. It doesn't even stand up to the functionality of an HTML page with a basic FORM embedded in it. But we were told not to despair; Sun promised that a much better technology was in the works. We were then presented with Swing, which definitely looked the part. However, while academically a sound design, in reality, it's a dog-slow technology that suffers from far too much abstraction and object creation to be practical. Swing applications require a lot of memory and a lot of horsepower to get anywhere. I know there are some great examples of Swing applications but, 9 times out of 10, I'm shown Forte and other heavyweight IDEs, which conversely are great examples of how much stress Swing really does put on the desktop.

On many Java forums around the planet are some great analysis pieces on why Swing is slow and impractical. I urge you to check out some of the discussions on Javalobby and Slashdot; you'll notice the conspicous absence of any counterarguments.

I've already mentioned in previous editorials how we've missed the boat with applets and how Macromedia has stolen Java's thunder with its Flash technology. It grates on my nerves every time I surf to a particularly Flash-heavy site, as this was the precise role Java had promised to fulfill. I strongly believe that AWT lost us this battle, and Swing was far too heavy to be anywhere near a browser to even be considered in the running.

The title of this editorial (sadly not mine but from a poster to our "Ask JDJ" feature, www.n-ary.com/java/jdj/askjdj_question.cfm?JDJ_ID=4) sums it up very well: if Swing is our only hope, boy are we in trouble!

However, don't despair. I have news…great news…about Java on the client side. We may have lost the first round, but the war is still being fought. Since we can't rely on Sun to offer help here (I'm still waiting for a return e-mail from Sun's tech lead for client-side Java, who still has to defend Sun's Swing position), we have to turn to the efforts of third parties – you won't be disappointed.

The first technology I'd love to tell you about is thinlets (www.thinlet.com) – an effort from an individual buried deep in Europe. This is a lightweight GUI framework with rich widgets that you can easily use. You describe the front end using XML, and it's quick and easy to build up an applet. The framework is 30KB. That's it! Check out the Amazon demo and please e-mail me if you aren't impressed. This is why Java is so exciting and shouldn't be underestimated.

The second technology is the Standard Widget Toolkit (SWT) from the Eclipse project (www.eclipse.org). This is a little heavier than the thinlet technology and builds applications that would ordinarily be installed on the desktop. It's a direct competitor to Swing and, from all reports, does a much better job. By all accounts, running an Eclipse IDE on a 400MHz machine is very fast, in stark contrast to running an equivalent Swing application within the same constraints.

If you need your faith in Java reenergized, look to the community and what they're doing. If you're waiting for Sun, don't. They're too busy fighting Microsoft to worry about the actual technology they're supposedly fighting for!

alan@sys-con.com

AUTHOR BIO

*Alan Williamson is the editor-in-chief of* Java Developer's Journal. *When not answering your e-mails and working on the next issue of* JDJ, *he heads up a small team dubbed the "Thunderbirds of the Java industry," providing on- and offsite rescue for Java projects in trouble. For more information visit www.javaSOS.com. You can also read his blog: http://alan.n-ary.net.*

SYS-CON MEDIA

PUBLISHER, PRESIDENT, AND CEO
FUAT A. KIRCAALI fuat@sys-con.com
COO/CFO
MARK HARABEDIAN mark@sys-con.com
VICE PRESIDENT, BUSINESS DEVELOPMENT
GRISHA DAVIDA grisha@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 MANAGERS
CARRIE GEBERT carrieg@sys-con.com
KRISTIN KUHNLE kristen@sys-con.com
ALISA CATALANO alisa@sys-con.com
LEAH HITTMAN leah@sys-con.com
EDITORIAL
EXECUTIVE EDITOR
NANCY VALENTINE nancy@sys-con.com
EDITOR
M'LOU PINKHAM mpinkham@sys-con.com
MANAGING EDITOR
CHERYL VAN SISE cheryl@sys-con.com
ASSOCIATE EDITORS
JAMIE MATUSOW jamie@sys-con.com
GAIL SCHULTZ gail@sys-con.com
JEAN CASSIDY jean@sys-con.com
ASSISTANT EDITOR
JENNIFER STILLEY jennifier@sys-con.com
ONLINE EDITOR
LIN GOETZ lin@sys-con.com
PRODUCTION
VICE PRESIDENT, PRODUCTION AND DESIGN
JIM MORGAN jim@sys-con.com
LEAD DESIGNER
LOUIS F. CUFFARI louis@sys-con.com
ART DIRECTOR
ALEX BOTERO alex@sys-con.com
ASSOCIATE ART DIRECTORS
CATHRYN BURAK cathyb@sys-con.com
RICHARD SILVERBERG richards@sys-con.com
ASSISTANT ART DIRECTOR
TAMI BEATTY tami@sys-con.com
WEB SERVICES
WEBMASTER
ROBERT DIAMOND robert@sys-con.com
WEB DESIGNERS
STEPHEN KILMURRAY stephen@sys-con.com
CHRISTOPHER CROCE chris@sys-con.com
CATALIN STANCESCU catalin@sys-con.com
ACCOUNTING
ASSISTANT CONTROLLER
JUDITH CALNAN judith@sys-con.com
ACCOUNTS RECEIVABLE/COLLECTIONS SUPERVISOR
KERRI VON ACHEN kerri@sys-con.com
ACCOUNTS PAYABLE
JOAN LAROSE joan@sys-con.com
ACCOUNTING CLERK
BETTY WHITE betty@sys-con.com
SYS-CON EVENTS
VICE PRESIDENT, SYS-CON EVENTS
CATHY WALTERS cathyw@sys-con.com
CONFERENCE MANAGER
MICHAEL LYNCH mike@sys-con.com
SALES EXECUTIVES, EXHIBITS
MICHAEL PESICK michael@sys-con.com
RICHARD ANDERSON richard@sys-con.com
CUSTOMER RELATIONS
CUSTOMER SERVICE MANAGER
ANTHONY D. SPITZER tony@sys-con.com
CUSTOMER SERVICE REPRESENTATIVE
MARGIE DOWNS margie@sys-con.com
JDJ STORE MANAGER
RACHEL McGOURAN rachel@sys-con.com

# Is Complexity Hurting Java?

WRITTEN BY JASON R. WEISS

EJB. JSP. JMS. JMX. JCA. JTA. JAAS. JAXP. JDBC. JNDI. This is a partial list of the acronyms you'll find in the 228-page J2EE v1.4 public draft. Of course, I was able to assemble this list of acronyms before I reached the bottom of page six.

This new version of J2EE is presented as the Web services version, so to be fair I should add SOAP, SAAJ, JAX-RPC, and JAXR to this list. In this age of integration, I wouldn't feel right if I didn't include JCA and JACC. I also feel obligated to at least mention the unfortunate specifications that were absent the day acronyms were handed out, like servlets, JavaMail, J2EE management, and J2EE deployment.

If this list of acronyms overwhelms you, I apologize. I do have a suggested course of action that will help you become educated in J2EE 1.4. Each one of the aforementioned acronyms is a specification unto itself, so all you have to do is read each one, and you'll be set! Let's see here…the EJB 2.1 PFD specification is only 640 pages, so we can cover that on Monday and Tuesday. On Wednesday we can review the Servlet 2.4 PFD specification, which is a more palatable 307 pages. Then Thursday we can download and review the JSP 2.0 PFD specification – a mere 374 pages. Hmmm, maybe reading each of these in sequence isn't a solution either, unless you're fortunate enough to be employed by a company that will let you sit and read specifications for a month straight (in which case, where do I apply for a job!).

We must ask ourselves what makes a technology successful. Looking back about 12 years, we saw the popular adoption of client/server technologies, and we saw business solutions implemented rather quickly. The kingpins of client/server were PowerBuilder and Visual Basic. I was fortunate enough at the time to be a Microsoft Certified Trainer and a Certified PowerBuilder Instructor. In the course of about five years, I taught a little over 1,000 students, balanced with intermittent consulting engagements that kept my feet in the real world. While client/server had its weaknesses, I think we can reach a consensus that by and far PB and VB were successful at what they did. In fact, both technologies passed Wiseman's Law, which states:

*A successful technology will saturate an 80% sampling of programmers only if 80% of the technology can be understood by those same programmers without forcing them to work beyond their regular work hours.*

Rarely is there a discussion on the learning curve of a piece of technology. As a manager I'm responsible for a team of programmers and I must ensure that the team maintains a high degree of productivity. One weakness of Wiseman's Law is that it doesn't define any time frame for achieving the 80% milestone.

What is an acceptable time frame for a learning curve? Again, let's look at history. Both PB and VB offered a one week fast-track training course. A developer certainly wasn't ready to pass any certification tests after that one week. However, most students were fluent enough in the technology that after completing the course, assuming they used the technology on a daily basis at work, within three months they could be implementing business solutions.

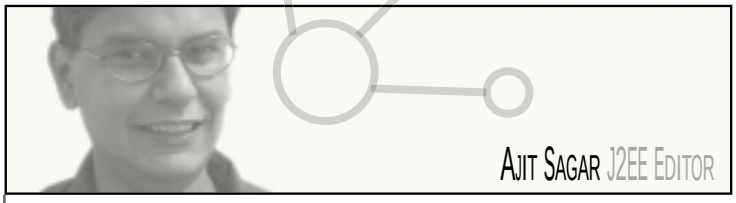
If we applied these same metrics to Java and J2EE, how would the technology rate? Right now, I believe the answer is very poorly. The problem, however, lies not with Java or J2EE, but with the way Java evolves through the Java Community Process. The formal goals of the JCP are found at www.jcp.org:

***–continued on page 46***

jason@getpersonified.com

AUTHOR BIO

*Jason Weiss is the director of technology at Personified Technologies, LLC, where he spends his days architecting the company's flagship product, GetEngaged.*



AJIT SAGAR J2EE EDITOR

# Complex Relationships

EJB 2.0 is testimony to the fact that the J2EE model has come a long way. You can do a lot of things with 2.0 that were tedious and error-prone in EJB 1.1. The Container Managed Persistence (CMP) relationship management alone makes it worthwhile. Just define all database access through method calls on entity beans, configure the deployment descriptors to recognize the method calls, set the autogeneration of RDBMS tables to "true," start the server, and you're ready to rumble. No more mucking around with databases. Once your DBA has set up your relational database, you never have to look at it again. As far as your application is concerned, you deal in objects, whole objects, and nothing but objects. Right?

Wrong! While the EJB 2.0 architecture definitely alleviates the pain of dealing with databases directly, it cannot completely eliminate it. Note that I'm not making light of database maintenance. On the contrary, the main point of this editorial is that the moment you use more than what the EJB model offers, you need to make sure your project has a person who can synchronize and coordinate between the object and relational worlds. When you get around to building an application with complex relationships between entities, there are several factors you need to consider from a programming perspective. Relationships between entity beans are handled quite cleanly by CMP, as are cascading deletes – at least simple cascades.

However, unless you create foolproof code that perfectly manages data integrity on the first shot, which you may in a small-to-medium application, you're bound to get dirty data in your database because of logical errors. By dirty data I mean data that doesn't necessarily conform to the entity relationships the way you had intended. Then comes the data cleansing stage, which can cascade into multiple problems at several levels. This is because the data cleansing at the RDBMS level may be out of sync with the cleansing required by the object model.

The alternative is to identify the appropriate constraints in the database that restrict the entry of inconsistent data from the object model. If you were programming in JDBC directly, chances are the constraints would have been a part of your design from the get-go. However, the abstraction provided by the EJB model lets developers unfamiliar with the intricacies of databases program complex applications. A developer can more or less exist in his or her OO world without worrying about low-level SQL queries and mundane database tasks. However, when things go sour, it's very hard even for a seasoned DBA to come in and fix things.

The correct approach for a successful project is to make sure you have at least one person on your team who can deal with these issues at the same time your EJB developers are designing their object components. In some cases the RDBMS constraints may conflict with the object model, and compromises may have to be made at either end. However, the end result will be that you will have a more robust and maintainable application. A fringe benefit is that you'll learn about how entity beans work behind the scenes and how to use them judiciously. Luckily, in our last project we were able to recruit a very knowledgeable developer who was familiar with both the object and relational worlds. Although my team started out designing from the object perspective, and all of us had some experience in RDBMS applications, his contribution made the difference between succeeding and failing in a large-scale J2EE project.

ajit@sys-con.com

AUTHOR BIO

*Ajit Sagar is the J2EE editor of JDJ and the founding editor of XML-Journal. He is the director of engineering at Controlling Factor, a leading B2B software solutions firm based in Dallas, and is well versed in Java, Web, and XML technologies.*

## J2EE INDEX

# J2EE FAQ

## IS J# A PART OF J2EE?

Absolutely not! J# is a language developed by Microsoft. It's very similar to Java in syntax and semantics. However, that's where the similarity ends. Java can be compiled into bytecodes (class files) that run on any platform. J# can only be compiled into code that runs on Microsoft Windows platforms. J# is a part of Microsoft's .NET initiative.

## WHAT ARE J2EE DESIGN PATTERNS? ARE THERE ANY GOOD SOURCES REGARDING THEM?

If you're familiar with the term *design patterns* as it applies to object-oriented programming, you'll find that EJB design patterns apply to distributed Java programming using EJBs and other J2EE APIs. J2EE design patterns describe patterns of solutions applied to similar problems in the same context.

Some good sources for EJB and J2EE design patterns are *EJB Design Patterns* by Floyd Marinescu; *Core J2EE Patterns* by Deepak Alur, John Crupi, and Dan Malks; http://developer.java.sun.com/developer/technicalArticles/J2EE/despat; and www.theserverside.com/patterns/index.jsp.

## DOES AN ENTITY BEAN ALWAYS NEED A PRIMARY KEY?

Yes. An entity bean represents a row in a table in a relational database. Each row in any RDBMS requires a primary key, which is the way to store and retrieve the data stored in that row. Since an entity bean maps to a row, it needs a primary key. The columns of the row (or the combination of fields of the entity bean) that compose the primary key are defined in the EJB's deployment descriptor.

## ARE THERE ANY OFFICIAL APIS FOR DEVELOPING WEB SERVICES IN JAVA?

While there are different tools and app server vendors that give you the ability to convert your Java components to Web services, Sun (the official source) offers a set of APIs for Web services in Java in the form of a toolset – the Java Web Services Developer Pack (Java WSDP). Java WSDP allows developers to build, test, and deploy XML applications, Web services, and Web applications on the Java platform. The Java WSDP provides Java standard implementations of existing key Web services standards, including WSDL, SOAP, ebXML, and UDDI, as well as important Java standard implementations for Web application development, such as JSP and the JSTL (JSP Standard Tag Library).

## WHAT IS EJB QL? HOW IS IT USED?

Enterprise JavaBeans Query Language attempts to fill in the gaps between the EJB model, which offers a standard way to access the database through the object model, and SQL, which offers a much richer set of queries for accessing RDBMS. EJB QL defines queries for the finder and for select methods of an entity bean with container-managed persistence. EJB QL is a subset of SQL92 and offers extensions that allow navigation over the relationships defined in an entity bean's abstract schema.

EJB QL is not used inside the EJB classes. Rather, the queries are defined in the deployment descriptor of the entity bean. The application server deployment tool translates these queries into the corresponding SQL that allows access to the database. The syntax of EJB QL is pretty much the same as SQL syntax and should be easy to pick up for someone familiar with SQL.

## THERE SEEMS TO BE A LOT OF UML DESIGN IN EJB. ARE THERE ANY APIS THAT ALLOW UML DESIGN TO TRANSLATE TO EJBS?

Some of the IDE vendors such as Borland (JBuilder) and Rational offer some capabilities to convert from one form to another. However, the conversion is usually one-way and not always maintainable as the design evolves. Currently there's a JSR 26 (Java Specification Request) called "UML/EJB Mapping Specification" that describes a standard mapping between the EJB architecture and UML.

## 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

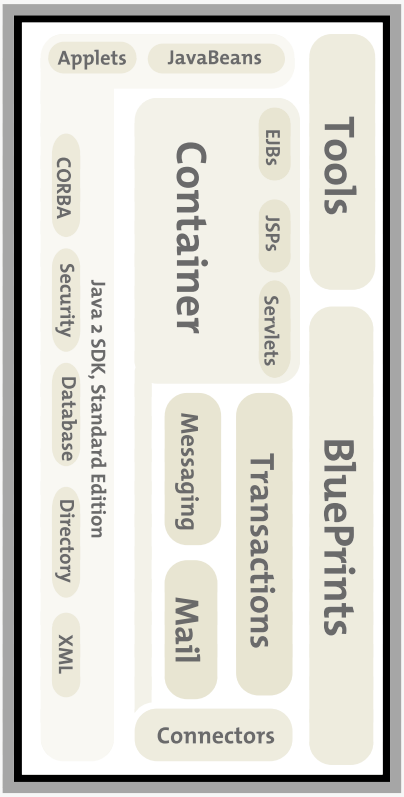
**Useful URLs:**

Java 2 Platform, Enterprise Edition
**www.java.sun.com/j2ee**

J2EE Blueprints
**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**



# JSR 168: Standardizing the Power of Portals

## Coupling portlets to portals

WRITTEN BY
STUART C. WELLS

**Recently, the first formal request for a new standard concerning the API between portals and portlets was submitted to the Java Community Process for review.**

Java Specification Request (JSR) 168 proposes a standardized interface that allows so-called portlets to be written in a way that makes them universally applicable across all portals from all vendors. Once realized, JSR 168 will have a far-reaching impact and will represent a significant step forward in defining portlets as a prime interface to Web services, which can be conveniently aggregated and personalized within portals.

To understand the long-term consequences of JSR 168, it's useful to review the role of both Web services and Web-based portals in the development of application software targeted at the Internet. The concept of Web services has energized the realm of application development by providing a new type of Web-based platform that connects users and widely distributed applications residing on servers. Web services are generally defined as modular, encapsulated functions that can be loosely coupled to provide specific results to an end user – either an individual, an application, or even another Web service. The continuing growth of the Web has created the potential for precipitous growth in Web services and a nearly infinite combination of interactions between services and users of all types.

But in another sense, Web services has created an embarrassment of riches for IT professionals. Without some way to anchor and focus Web services for the benefit of specific end users, their productive impact will be severely constrained. A very practical solution to this problem is found in the Web-based portal, a well-established entity that aggregates and organizes multiple streams of information produced by applications distributed across the Web. The recent profusion of enterprise-level portals has already demonstrated their value as an organizing framework for Web services at all scales of operation, from end users to departments and enterprises. In every case they bring coherence to a potentially overwhelming and chaotic flow of information.

While the portal represents the client side of solving the Web services problem, it obviously requires a complementary entity on the server side. This need is now being filled by a software component called the *portlet*, which owes much of its technical ancestry to the servlet.

In earlier phases in the evolution of the Java Virtual Machine (JVM), the original intent was to have platform-independent code that would execute on the client side in the form of applets. However, over time, the concept of platform-independent Java execution migrated onto the server side, where the majority of an application's execution occurs, leaving the client to act more in an interface role. Hence, the emergence of servlets.

A portlet further extends this concept by having a Java-based application execute on the server side and then present the user with a highly flexible interface that resides within the graphic real estate of a portal, which treats the application as one of many interactive channels presented to the user at any given time.

The coupling of portlets to portals, which act as organizers/aggregators of portlet content, offers significant potential to the developer community, technically and economically. From a technical standpoint, the portal represents a good generalized software framework to guide the course of content development. Its infrastructure provides a number of attributes that support the tactical details of content delivery and presentation. These include items such as single sign-on capability; flexible and extensible authentication; management of policies via identity; device identification; distributed session management; and the ability to deliver content across a wide array of end user devices, such as desktop machines, cell phones, PDAs, and kiosks.

From an economic standpoint, the portal represents a convenient and clear way to organize the market for portlet content. Enterprises, organizations, and

"With the API specified by the proposed standard, portlets would have well-structured, predictable access to their host portal environment"

service providers will use the portal as a highly flexible vehicle to package services in ways that represent the specific requirements of various user communities.

However, to realize the full potential of portals and portlets, one last barrier remains, and that's where the proposed standard comes in. While the general concept of a portal is well understood and accepted, this belies a great deal of diversity in the way portals are technically constructed and how they interface with portlets. Consequently, developers must currently pick a target portal and write code aimed specifically at the execution environment defined by that particular one. JSR 168 seeks to remedy this situation by defining an API that universally dictates the interface between all portals and portlets across the Web.

While there are numerous details within JSR 168, it's worth considering some of the highlights that show the potential benefit of the proposed API. Most fundamental, with the formalization of the portlet API, the specification request officially defines a new atomic unit of execution within the Java arena, in the same way that JavaBeans, applets, and servlets are recognized. This concrete definition of a portlet gives programmers a clear frame of reference to guide their programming efforts. At the same time, it provides a client-agnostic programming interface that extends across all end-user devices with full provisions for declarative security and remote execution.

With the API specified by the proposed standard, portlets would have well-structured, predictable access to their host portal environment. For instance, they would be able to obtain and act upon user profile information provided by the portal. They would also be able to dynamically interact with the portal window and action event model to make the best use of their allotted real estate on the portal page at any given moment. They would also have a stable model for managing configuration (personalization) on a per-user basis. At a higher level, the API would allow portlets to interact with other portlets within the same portal environment through data sharing.

For a standard such as this one to gain broad acceptance, it must deliver benefits that extend through the entire value chain of the Web services industry, and a quick tour through this chain shows that JSR 168 will have consistently positive impact. Upstream, at the IDE level, toolmakers will have an opportunity to expand and remarket their current offerings to embrace portlets as a new atomic unit of execution. For instance, with a standardized API, it will become possible to automatically derive portlets from other applications via a new type of IDE tool.

Further downstream, developers will have a significantly expanded market opportunity because their applications will be freed from the confines of specific portal requirements. Writing to these ever-changing interfaces consumes a large share of development dollars that could otherwise be put into new content.

Once new content becomes available, portal providers will have a wide range of options in configuring the information flow on their portals. Right now, providers must individually construct and maintain the interface for each service occupying space in their portal environment. With JSR 168 enacted, the standardized interface will substantially reduce the time and cost of integrating and supporting Web services packaged as portlets.

At the enterprise level, the proposed standard promises to streamline the process of delivering specific applications to users throughout the organization. Once the standard is in place, all application vendors will have a single target for portlet development, and can routinely provide a standards-compliant portlet as part of the application itself. Also, the process of deploying portlets will be greatly simplified and can easily span portal products from multiple vendors, if necessary.

Perhaps the biggest benefit comes at the far end of the chain for the user accessing a portal from a desktop or mobile device. When JSR 168 is fully implemented, portal users everywhere will finally realize the full benefit and richness of the portlet as a primary point of entry into the domain of Web services.

JSR 168 operates in the great tradition of technology standards that shift market competition from proprietary lock-ins to a contest based on the best implementation for the broadest set of users. While it may produce some short-term disruptions, the long-term result is a major win for everyone involved.

### ON PORTALS AND PORTLETS

Over the past couple of years, portals have come to play a key role in the delivery of applications, content, and, more recently, Web services. In general, a portal can be defined as a highly personalized Web site providing secure access to applications and business processes that the portal user shares with well-defined user communities on both sides of the firewall. All services delivered to the portal are based on a set of roles, permissions, and preferences assigned to the individual portal user.

For example, an engineer working on mechanical systems might have a much different portal configuration than an accountant assigned to inventory control within the same enterprise. Also, users have the latitude to extend beyond their portal's initial provisioning of applications and data and personalize its environment to suit their particular working style. At the same time, the portal should be able to support a full range of client device types and even different physical locations for the user. In terms of IT infrastructure, portals must be integrated with a number of other software components, including directory servers for user management and application or Web servers that provide a conduit to applications, data, reports, and transactions.

Portlets, on the other hand, connect to and display applications of various sizes and scopes that execute independently from the portal, but use the portal as either a dynamic or static interface to the user. In effect, portals can be thought of as collections of portlets. The permission to use these portlets is defined by the user's role in a given community, and is managed by the portal's interaction with various directory services. Multiple portlets may be running – and even interacting – within a user's portal at any given time.

**AUTHOR BIO**

*Stuart C. Wells is senior vice president of iPlanet Products and is responsible for the iPlanet product development and product marketing functions. Stuart holds a BS in electrical and electronic engineering, an MS in digital techniques, and a PhD. in video and image compression, all from Heriot-Watt University in Edinburgh, Scotland. Stuart also earned an MBA from Santa Clara University, California, and attended Stanford University's Executive Business Program.*


# Clustering J2EE Applications

written by Ashok Banerjee, Ganesh Kondal, & Sunil Kunisetty

Internet commerce demands 24x7 Web site availability. We need systems that provide zero planned downtime, so during routine maintenance, system upgrades, etc., the system can still serve client requests. We also need systems with zero unplanned downtime – one application server crashing or one computer accidentally turned off should not stop the system as a whole from working. We need to be able to handle far more concurrent requests than one server can handle. To do all this we need clustering.

These objectives cannot be met by strict adherence to J2EE APIs. It's not something the application does, but something the container or infrastructure provides.

A cluster is a set of server nodes that cooperates to provide a more scalable and fault-tolerant server infrastructure for stateful and stateless components. To external clients, a cluster appears as a single server that services requests with a single point of entry.

In this article we'll discuss clustering with regard to the J2EE platform, focusing on the Web tier. We'll discuss the various tiers of clustering, the mechanisms, and their performance/scalability ramifications, and illustrate how to design highly available, scalable, fault-tolerant, and performant systems.

> *One minute of system downtime can cost an organization anywhere from $2,500 to $10,000 per minute. Using that metric, even 99.9 data availability can cost a company $5 million a year.*
>
> —The Standish Group

## Tiers of J2EE Clustering

Clustering is not limited to any single tier. It can be achieved in every tier – cache, Web (servlet, JSP), EJB, and even at the database (see Figure 1).

Caches sit in front of Web servers caching Web content and acting as virtual servers. They reduce the response time, offload the back-end servers, and help the Web tier handle the huge volume of client requests, thus increasing scalability.

The Web tier sits behind the cache layer. In a typical J2EE solution, the Web tier is made up of servlets and JSPs that are responsible for dynamic content generation.

Beyond the Web server tier is the EJB tier that runs stateless session beans, stateful session beans, and entity beans.

Even the last layer of the database can now be clustered using a database cluster, as with Oracle9*i* Real Application Clusters.

## Aspects/Facets of J2EE Clustering

The two major aspects of clustering are:

1. Load balancing or front ending
2. Fault tolerance or reliability

## Load Balancing or Front Ending

The idea behind load balancing is to distribute the load (from client requests) to multiple back ends. This enables a cluster (collection of cooperating servers) to handle more requests/clients than an individual server, thus providing scalability. A load balancer sits in front of the server nodes and receives requests, then redirects these client requests to the various back ends.

- This distribution of the workload boosts performance (shorter response time).
- Individual servers in the cluster can go offline for maintenance without causing the system to halt or fail (zero planned downtime).
- This provides scalability because the cluster can handle more clients than any individual server.

### *Group Membership*

Server nodes need to be registered with the load balancer to have requests routed to them from the load balancer (see Figure 2). These registrations can happen statically or dynamically.

- In static registration the load balancer is configured statically beforehand with information about its target list of servers to register. To change this list, the load balancer must be restarted.
- In dynamic registration, server nodes register dynamically at runtime with their target load balancer. This enables you to add new servers to the mix dynamically without bringing down the load balancer.

Usually the dynamic registration is done using bidirectional notification at startup.

Server nodes notify the load balancer at startup and register with the target front end. The load balancer in turn lets all potential servers know when it comes up so that server nodes seeking to register with it can do so. The load balancer is the

single point of entry to a cluster and this becomes a single point of failure.

To add reliability to the load balancer and ensure availability, a process monitor daemon can ping the load balancer; if the load balancer does not respond, the daemon can restart it. The load balancer can be behind a hardware load balancer that's using a virtual IP. To prevent a restart from interrupting the request routing, it's important for the load balancer to be stateless. In other words, all routing information for the session is transferred as a cookie with each request.

### Load Balancing Strategies

- ***Random:*** As the name indicates, the requests get dispatched to back ends in a random manner.
- ***Round-robin:*** Refers to a distribution on a round-robin basis. This can result in a bad situation: if we have two servers and our request pattern is a big heavy workload request followed by a light workload request. In a round-robin situation, all heavy requests go to one back end and all lightweight requests go to another.
- ***Weighted:*** Any of the algorithms can have weights associated with them. For example, I want twice as many requests dispatched to server A as to server B. This implies a weight of 2/3 : 1/3 for A:B.
- ***Equal request:*** Load balancer maintains a counter for each node ensuring all nodes receive an equal number of requests (or weighted).
- ***Equal client:*** Requests get dispatched based on the client requesting it. This proves useful for stateful sessions. The load balancer sticks a session to a server node, which means all requests from a given session will be handled by the same server node (in failover cases this will hold until the server fails).
- ***Equal workload:*** The load balancer keeps track of the workload handled by each server node by polling them at regular intervals, and weights can be adjusted for distribution. So workload statistics, if fed to an adaptive weighted system on a load balancer, can load balance an equal workload.

Even though this may seem the ideal way of load balancing, the overhead of calculating the workload results in performance hits that may make it undesirable.

## Failover and Reliability

When an application server in a cluster fails to serve client requests, the load balancer reroutes the requests to its peer(s). This is termed as *failover* and provides the basis for fault tolerance and reliability in the cluster. Each server node of a cluster names one of its peer as its secondary server node. If a server node fails, the load balancer finds its secondary/secondaries and reroutes the requests.

The advantages of failover are:

- User is immune to system crashes
- Reliability

The disadvantages of failover are:

- Redundancy
- Performance hit due to replication

### Replication

Failover is achieved through replication. Replication is done using either:

- Point-to-point using direct socket communication
- Multicast the message to the entire group

Stateless session failover is tantamount to a simple request redirection. Stateful session failover on the other hand requires both request redirection and session replication.

Replication is a two-step process of transmission and consumption. Transmission from the sending server VM occurs according to some strategy while message consumption on the receiving VM occurs lazily. The bytestream received is not deserialized until needed. Because it's not materialized into the HttpSession, stateful sessions tend to be sticky. We see that the entire system is an *n* serialize (serialization happens many times on the transmitting side), or a 0 or 1 deserialize (message is consumed lazily only if needed, i.e., in case of a failover) system. Stickiness of sessions is done using the jsession_id cookie.

### Replication Methods

Replication can happen over different transports.

Point-to-point transport replication is typically done for a single primary and a single secondary. Here you typically use a TCP socket connection between the primary and a secondary server. In case of a failover, the client can actually failover to any node in the cluster. Thereafter the node receiving the request can request the session from the cluster and then the secondary will send the session and continue to serve as the secondary for the new primary.

An alternative to point-to-point replication is to replicate using UDP multicast. However, the primary concern with it is the reliability of the message delivery. TCP packets have an acknowledgment built into the system but, on top of UDP, application server vendors often build a NAK (negative acknowledgment). Each packet can be sequenced with an ID, and if the receiving VM finds that it received packets 1, 2, 3, and 5, then it knows it missed packet 4. UDP-based systems can have multiple secondaries or an island (as in Oracle9*i* Application Server) unlike point-to-point systems. Since UDP-based systems are not waiting for acknowledgment, the replication can happen asynchronously and so such a system can be more performant.

In this system, state replication happens within an island and load balancing happens within and across islands. However, once a stateful session is bound to an island, it should continue to work against the island. A stateful session once bound to a back end sets cookies that identify the back end as well as the island to which it was bound.

### Web Application Failover

Web application failover requires that:

- The application is marked distributable.

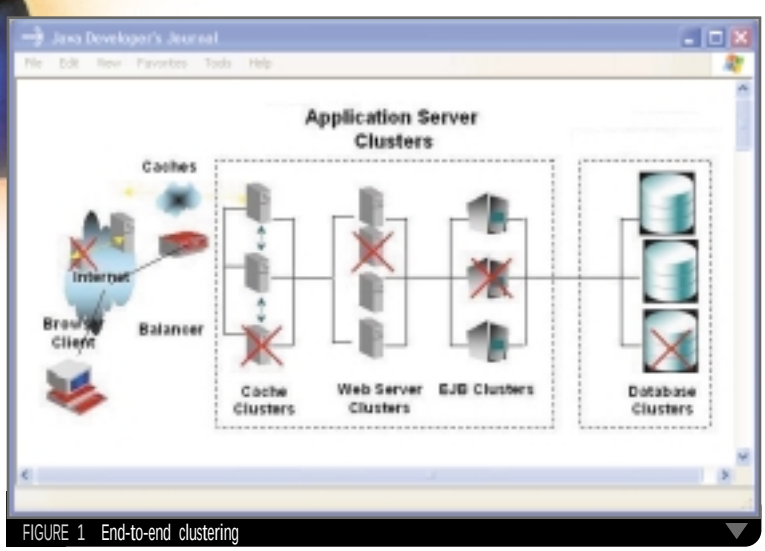



FIGURE 1 End-to-end clustering

- Objects in the HttpSession are serializable or remote.
- Instance/static variables should not store state. Such variables do not make sense when we failover to a different Java Virtual Machine (server node).
- ServletContext should not store state.
- EJBs/database should store long-lived application state.

```
public void doGet (…) … {

HttpSession session = request.getSession(true);
Cart cart = (Cart)session.getAttribute("cart");
if (cart == null) {
cart = new Cart();
session.setAttribute("mycart", cart);
} ….
```

In the Web clustering replication most application server vendors currently replicate when setAttribute/removeAttribute is called on HttpSession.

This implies that if we do a setAttribute for cart, then update it and don't setAttribute again, the secondary/secondaries won't have the updated state.

There are times when you may want to use load balancing without failover:

- In pure performance when we're not prepared to take the hit of replication
- When we have nonserializable objects in HttpSession
- When the requests are stateless

### *Common Pitfalls*

In general, failover is transparent to the end user except in some cases that I'll mention later.

Static variables in a servlet should not be used, but in most servlet containers users can get away with using static variables. However, in distributable applications, static variables will be reset in the new/failed-over VM and will therefore be different from the values in the original VM.

Most people assume that the init method of a servlet is executed only once. This is true in one VM; however, in a failover scenario the init method of a servlet executes once per failover (once on each VM).

Remember that in case of a failover, we go back to the last state in which the primary did a setAttribute for each attribute. If after certain updates a setAttribute was not done, we won't find those updates on the secondary/secondaries.

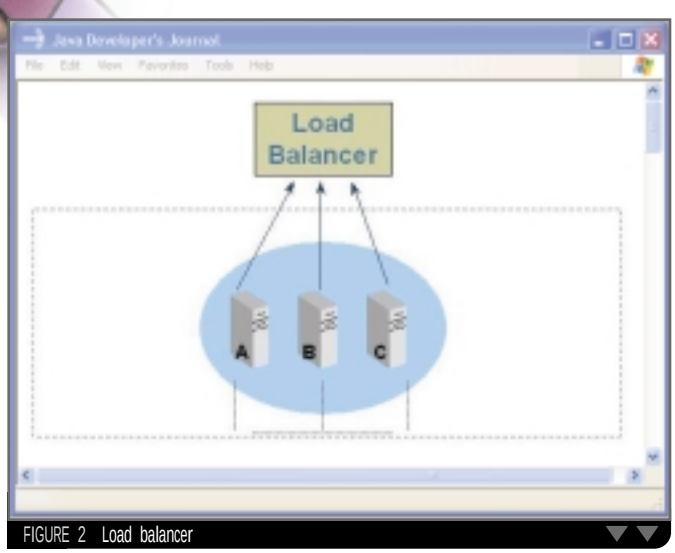



FIGURE 2 Load balancer

### *Performance Optimizations*

The price of plurality lies in replication and its related network overhead.

The system is not *n* serialize – *n* deserialize. Almost all clustering solutions use *n* serialize 1 deserialize, so anything to speed up serialization helps.

In the course of serialization a serialVersionUID computes the version of an object and this can be a time-consuming operation that's being undertaken for every serialization-SerialVersionUID. This is a private static long defined in a class.

Preferably, provide a readObject/writeObject for the class being serialized. This can save on time spent in reflection to get the attributes/state variables to serialize. When writing a point object the writeObject can just write the values of *x* and *y*; for the default serialization mechanisms to serialize an object of type point, it must reflect on the class Point and find out that *x* and *y* are the attributes to serialize followed by reflecting to obtain the values of *x* and *y*.

If a network is the constraining factor, it may be useful to find the difference between previously serialized streams and transmit it. However, this typically has the overhead of computing the delta or difference between the streams.

## Recommendations

- Multisect the problem space so that the secondary/secondaries do not have common causes of failure.
- Use as much redundancy as possible.
- Eliminate common causes of failure, such as a common source of power.
- For a session, preferably pick a secondary so that both primary and secondary are not on the same machine.

## Summary

In this article, we looked into clustering, focusing on the Web tier. We also discussed the two major aspects of clustering: load balancing and failover. To provide high availability, load balancing, scalability, and reliability, it's important to choose the correct application server infrastructure and configure it appropriately. An insight into the underlying mechanisms can prove invaluable. Application servers, such as BEA WebLogic and Oracle9*i*, provide clustering. In subsequent articles we'll discuss clustering in the EJB tier.


## Author Bios

*Ashok Banerjee is a technical lead at BEA working on WebLogic Server. Prior to this he worked at Oracle Corporation for four years on Oracle9iAS, and at IBM for two years. A regular presenter at JavaOne and other conferences, Ashok also teaches Java design patterns at the San Jose State University.*

*Ganesh Kondal is a senior engineer at Modulant Solutions. He has been working on distributed computing and Java for the past four years and is now moving into Web services and Linux. His interests are distributed computing, Web services, and design patterns. Ganesh is currently working on a master's degree in science and engineering at San Jose State University.*

*Sunil Kunisetty is a technical lead and lead developer for Oracle Web Services. He's the expert member of JSR 109 (Enterprise Web Services for J2EE) and WS-I Basic Profile communities from Oracle. He has over six years of distributed computing experience with an emphasis on Java. Sunil is a frequent speaker at Oracle OpenWorld, JavaOne, and Oracle iDevelop conferences.*

ashok.banerjee@bea.com

gkondal@modulant.com

sunil.kunisetty@oracle.com

# External SC Architecture and VO Cache

## Boosting the performance of enterprise applications

WRITTEN BY
MIKHAIL SKORIK

**In this article I introduce two practical software design techniques that are useful in the development of high-performance, horizontally scalable, and immediately cost-effective enterprise solutions built on J2EE technology.**

I'll be using a distributed enterprise application that runs in a J2EE application server. The first technique is in designing the overall system following the external servlet containers (external SCs) architectural approach. This aims to increase the scalability value of an enterprise application and reduce the cost of an overall solution. The other technique is the value object caching mechanism (VO cache) with an integrated JMS-based update tracking facility. The purpose of this is to compensate performance penalties caused by moving presentation logic out of the application server (AS) process, and to increase performance by eliminating unnecessary communication with the underlying business and persistent storage layers. A key feature of the proposed VO cache implementation is the ability to synchronize cache contents with the underlying persistent storage without needing any timeouts or refreshes. Both techniques, combined, will help you to dramatically increase the performance of your enterprise applications.

The modern technique for achieving higher performance is designing an enterprise application that is distributable and horizontally scalable. But sizing an application in the traditional way by employing additional application server instances is costly since you have to procure more expensive licenses. An alternative approach is the external SC architecture that also allows the presentation logic layer to be scaled independently. This technique involves the delegating of presentation data formatting to a number of out-of-process SC instances run on separate physical machines. Since the licensing cost of an SC engine is a lot lower than one of a full-featured application server – you could end up saving thousands of dollars by scaling solutions this way.

In many enterprise applications it's typical for the number of read accesses to vastly exceed the number of writes. Usually developers implement some caching mechanism on the client side (or at the presentation tier) of an enterprise application to reduce the load on the underlying system tiers. However, ordinary caches don't guarantee that their content is really consistent with the data in a persistent storage. Instead, cache content is "considered" up-to-date for a limited time and is dropped or refreshed when this period elapses. The VO cache implementation is free of this disadvantage because it has the special built-in update tracking facility. Cache components are designed to be used as a private object on the client side of an enterprise application – for example, in an external SC instance or in a fat Java client application. The following is a summary of the benefits offered by an application built by applying both techniques:

- Data is always kept in sync with the underlying persistent storage.
- Client applications are provided with instant read-write data access.
- Separately scaled presentation- and business-logic tiers.
- Increased performance due to the enterprise bean VOs caching.
- Increased performance due to the business logic, consolidated into the more compact AS node.
- Reduced cost of the overall system.

### Target Application Architecture

First I want to introduce the concept of a target application – an enterprise application built according to solid design principles for effective leveraging of both proposed design techniques.

### Number of Isolated SCs at the Front End

At the highest level of abstraction, the architecture of the target application looks like a number of independent standalone SC instances run on their own physical machines and connecting to a dedicated AS node (single multi-CPU physical machine or cluster) over a network. In this approach, the only component doing real work in the AS node is the EJB container, not the SC. This approach relieves the application server from doing the work of presentation data formatting, thereby saving its CPU resources for effective servicing of EJB components. Thus, all data formatting is delegated to a number of less-expensive external SC engine instances. The number of nodes hosting SCs can be estimated using the following ratio:

> Cache components are designed to be used as a private object on the client side of an enterprise application

$$N \frac{DF\_web}{DF\_enterprise}$$

where DF_web is the average dataflow between the presentation tier and Web clients, and DF_enterprise is the average dataflow between the presentation- and business-logic tiers. Despite having better scalability value in comparison to traditional horizontal scaling, this approach isn't widely used because of one significant disadvantage: the increased number of RMI calls significantly impacts the performance when your system becomes distributed. I'll show how to overcome this by supplying external SCs with an entity bean state caching component.

## Private VO Caches

As shown in Figure 1, each SC instance holds its own private VO cache component. Since these components are allocated in-process with hosting SCs, and because of the extreme scalability of the presentation tier built with separate SC instances – all this results in dramatic overall system performance improvements.

## Shared O/R Mapping Components

The only modules that dedicated AS nodes run are the ones that contain O/R mapping and business logic. Mapping components are just the set of entity EJBs that handle the persistence of business domain (DOM) objects and ensure their integrity (cascade operations). The state holder objects of such entity beans will be further referred to as Domain VOs. Mapping EJBs are accessed (directly or by means of Session Façade components) by all kinds of client applications – from SC instances to ordinary fat Java clients. In addition, for a VO cache update-tracking subsystem to work properly, AS must have exclusive access to an underlying persistent storage – i.e., any data manipulations bypassing the O/R mapping layer are prohibited. It's similar to the common requirement when an AS is configured with commit option A (reusing a pooled EJB state, see J2EE spec for more info) and designated to avoid data inconsistency.

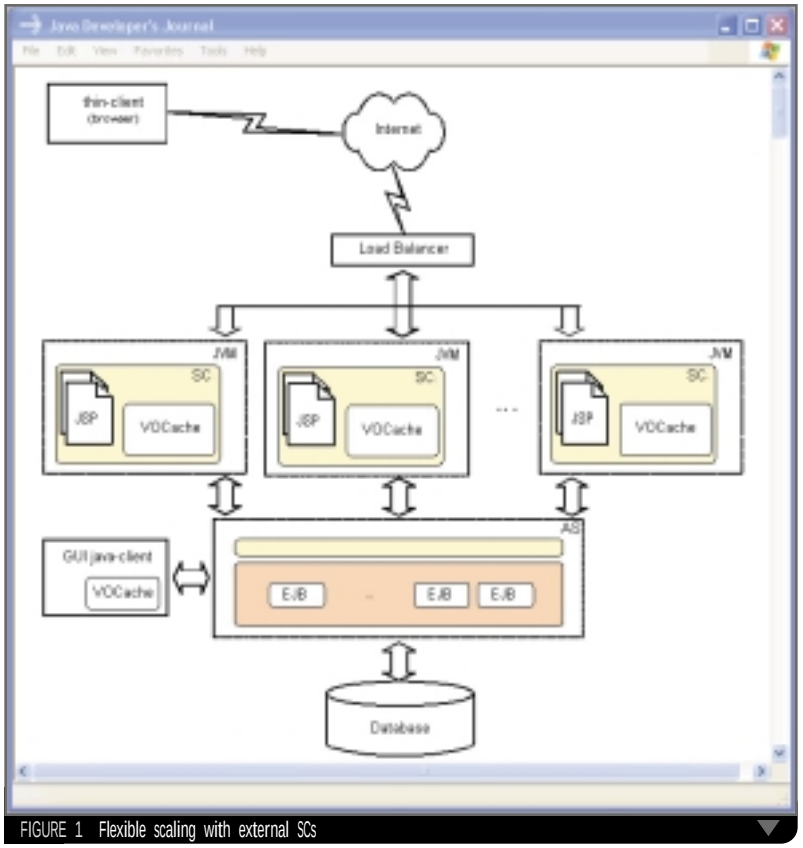



FIGURE 1 Flexible scaling with external SCs

## Caching Attribute VOs and Noncaching VO Lists

The VO cache implementation mentioned in the article was designed to operate solely on the Domain VOs. Although it's possible to adopt it to handle the Custom VOs (ones built according to the needs of the client application), a discussion on this topic is outside the scope of the article. Another issue to be addressed is the strategy for handling VO lists when the main entity (Title, for example) has associated attributes (Author and Publisher) related to it as one-to-many. There are two ideas involved here: obtaining the list of remote object IDs in one call (Bulk Retrieval Method or Fast-Lane Reader patterns) and further querying the attribute EJBs individually via a special cache-aware Proxy/Factory object. Attributes, which are often entities that don't span large amounts of space in persistent storage, are good candidates for caching. Finally, the client-side VO cache, built according to the principles described, will offer much more efficient caching of EJB states than a built-in AS instance pooling facility, since it eliminates the all unnecessary access to the business tier.

## On-the-Fly Assembling of Presentation Data

Since the VO cache operates solely on the Domain VOs, the human-readable form of data needs to be assembled on the fly at the presentation tier of an enterprise application – just before sending data to the end user (browser).

## VO Cache Reusable Library

### *Abstract O/R Mapping Components*

The abstract O/R mapping subsystem is a part of the library in which common interfaces of mapping components are defined, and EJB connection and VO cache lookup code is encapsulated. All the VOs are immutable. Any particular VO instance is in a one-to-one relationship with the corresponding Key instance (may be a composite). Intertier data transfer is made up of interchanging VO instances and of sending JMS messages. Figure 2 illustrates the data flow between major application components.

**AUTHOR BIO**

*Mikhail Skorik, a senior Java developer/architect at HPS International, has more than six years of development experience on a variety of platforms, the last three years in developing Web-servicing solutions for e-commerce. Mikhail holds an MS in applied mathematics from the Moscow Institute of Physics and Technology (MIPT).*

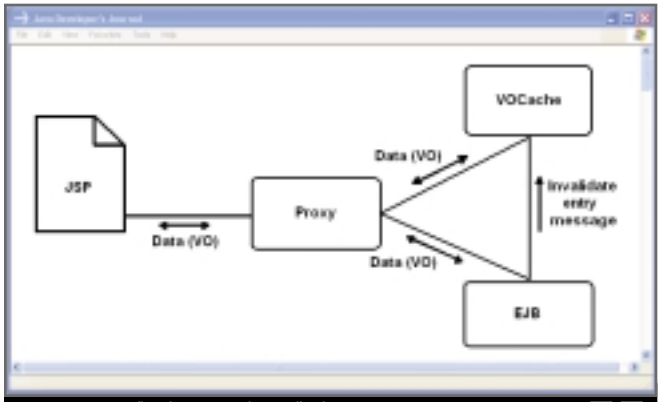



FIGURE 2 Data flow between major application components

A helper Proxy object was introduced to encapsulate the details of the O/R mapping component and VO cache interaction, hiding them from the client application programmer. A client application doesn't know anything about the presence of a VO cache, i.e., the Proxy acts like an object factory for VO instances. The state of the O/R mapping component may be updated either by accessing the EJB directly or through a Proxy object. Both cases are handled appropriately.

## VO Cache Core

The VO cache is an ordinary (not-enterprise) JavaBean that implements the Singleton pattern. This ensures that only one object instance resides in the memory of a JVM. The purpose of this component is to hold and manage VO/Key pairs. The Key object is used for cache lookup for a VO, as well as for locating remote EJB components. The Lazy-Load pattern was applied to handle the updates of cached entries. When an "invalidate" message occurs, the corresponding entry (VO instance) is just dropped without retrieving up-to-date data. A standard Java hashtable is employed to hold VO/Key pairs, and the FIFO queue is used to prevent cache overflow.

## JMS-Based Update Tracking Facility

Each VO cache instance has its own aggregated controller object, which is actually the implementation of a JMS topic listener. Similarly, every O/R mapping component has the aggregated JMS topic publisher, used to notify cache controllers about EJB state change. The only message topic residing inside an AS is used for communication. When the state of an O/R mapping component changes, all cache controllers receive the "invalidate entry" message. To allow multiple listeners and publishers to coexist in a system, the Publish/Subscribe model was applied. Note that JMS messages are always routed from an O/R mapping component to a VO cache.

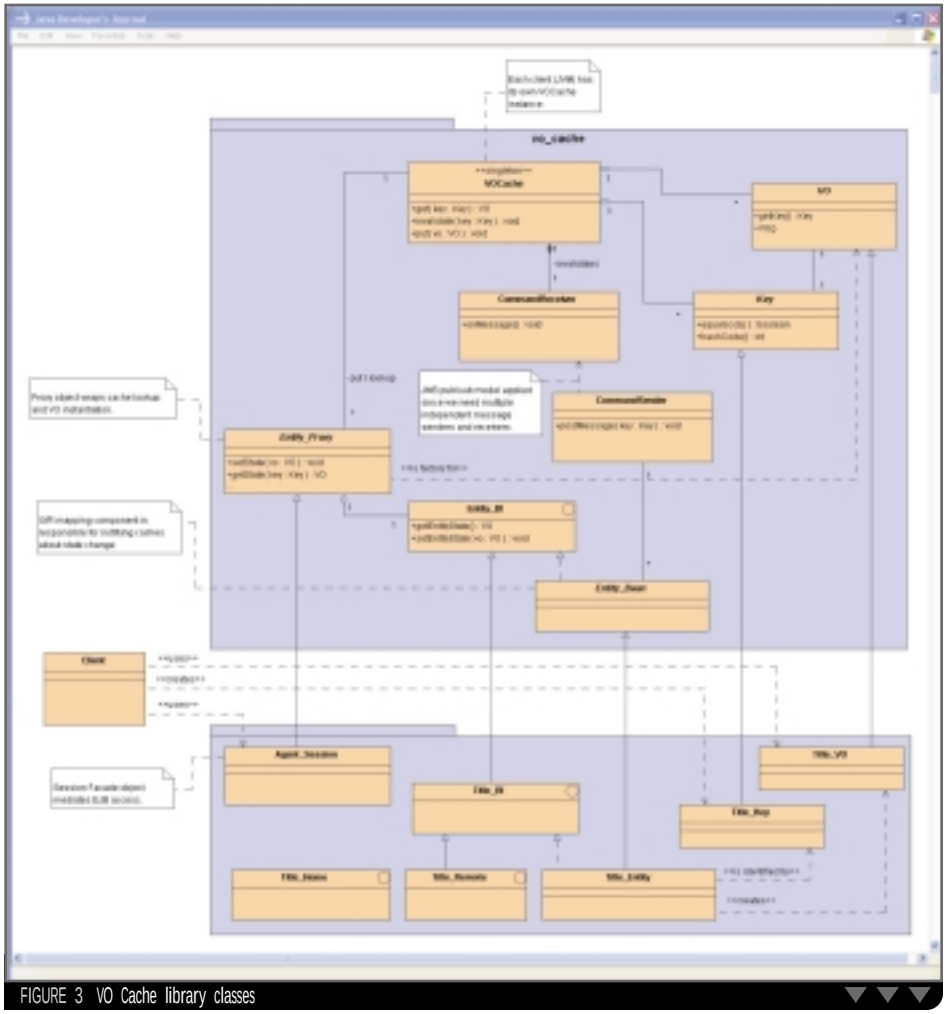


FIGURE 3 VO Cache library classes

### *Object Diagrams*

Figure 3 shows the objects that make up the reusable VO cache library and their subclasses, which the O/R mapping layer of the sample application is comprised of.

### *The Sample Application*

The design of the sample application conforms to the definition of the target application architecture mentioned earlier. Although the complexity of this application has been reduced for the purpose of this discussion, the design techniques mentioned apply equally to the real-world applications. The O/R mapping layer of the sample application consists of several bean-managed EJB components packaged into a single module. Note that the attribute VOs (Publisher, Author) of a main entity are retrieved from an O/R mapping layer individually, rather than the main entity VOs (Title), which are obtained as the preordered list. Data assembly for an end-user presentation is managed by the JSP pages. The sample application is implemented using the standard set of J2EE APIs without any proprietary AS vendor extensions. It's intended to run in an environment that's comprised of a dedicated JBoss application server node (may be a cluster) that connects to a number of standalone Tomcat instances run in separate processes.

## Performance Test

The effect of introducing the proposed design techniques was evaluated by comparing the performance values (as average response times) recorded while running sample applications in different configurations. There are three major factors contributing to the overall system performance. First, the penalties created by moving presentation logic into out-of-process SCs. Second, the performance gain from introducing the VO caching. And, third, the performance gain as the result of consolidating business logic into a more compact AS node (either a dedicated physical server or

cluster). Since, the third factor is difficult to estimate correctly because it requires special suits and a complex testing environment, I've limited my investigation by only using measurements from the first two. The total count of page retrievals in every test was 700 and the total count of records in the sample database was 1,000/500/100 (as titles/authors/publishers). For both tests JBoss was configured with a commit option A (reusing pooled EJB instance states). Notice that along with reusing pooled EJB states, the introduction of the VO Cache results in a dramatic performance improvement – more than 250%.

Figure 4 illustrates the smoothed average of performance dynamics while retrieving catalogue pages with random numbers. This proves that EJB state caching at the client-side may be much more efficient than delegating to AS facilities. Another thing to note is that cache contents are constantly in sync with an underlying database, meaning that all changes, made by other clients, are reflected instantly. (JMS is fast enough to be used for synchronization purposes at the presentation tier, although it isn't really synchronous.) By downloading source code from ***JDJ*** Web site, www.sys-con.com/java/sourcec.cfm, you can easily reproduce these tests.

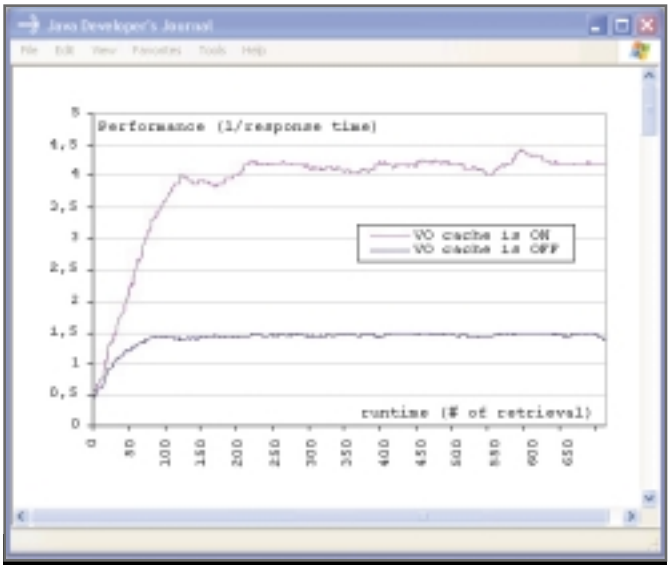


FIGURE 4 Performance test

## References


- Roman, E., et al. (2001). *Mastering Enterprise JavaBeans*. John Wiley & Sons.
- Marinescu, F., Roman, E. (2002). *EJB Design Patterns*. John Wiley & Sons.
- *The J2EE Tutorial:* http://java.sun.com/j2ee/download.html#tutorial
- *Java Developers Connection, J2EE Patterns Catalog:* http://developer.java.sun.com/developer/restricted/patterns/J2EEPatternsAtAGlance.html
- *Memoirs of eXtreme Dragon-Slayers, Part 18:* www-106.ibm.com/developerworks/ibm/library/i-extreme18
- *JBoss (unofficial) manual:* www.jboss.org/online-manual/HTML
- *JBoss-2.4.4 with embedded Tomcat-4.0.1:* www.jboss.org/downloads.jsp


mike_skorik@yahoo.com

JASON BELL J2SE EDITOR

# Success Without a Successor Is Failure!

I've always believed that we should pass on our knowledge to our peers, then, over time, we'd have a network of programmers who had a firm foundation in how Java works. To that end I try and help out where I can. I'm a member of the JDJList archives and I like to chip in once in a while. This is where all my troubles started.

Some people seem to think that if they demand a solution, they'll get it. Not so. Guitarist Robert Fripp used a wonderful phrase with his students many years ago: "The quality of the question determines the quality of the answer." I know it sounds harsh but it's true. It made the students think about the questions they were going to ask and phrase them in such a way as to get a meaningful answer back.

To all of you who ask questions on Java mailing lists, please take heart and accept that the people helping you have already walked the path you're walking. There's no shortcut; it's just a case of picking up the map and walking it yourself. Before you wade into a mailing list and ask a question that will provoke a response, think about the two questions below.

## Have You Read the API Docs?

You'd be surprised how many times people ask how to convert a string into an integer. My response is the same whether it's on JDJList, IRC, or in plain speech. Have you looked at the API docs? Alan told me a great line, "Write a man's code; he solves for a day. Show him the API docs; he solves for life!" The API docs are one of the most useful tools that Sun ever supplied; I just wish I had more time to read them. If you want a copy of the Javadoc for the Java language, visit the Sun Web site at http://java.sun.com/j2se/1.4/docs/tooldocs/windows/javadoc.html#commandlineexample.

## Have You Tried a Search Engine?

Chances are your problem is not unique and someone else has found a solution. I'm amazed how many people don't actually do a search on Google and type "converting a string to integer in Java." Lo and behold the first two results actually point to the API docs. Now isn't that a surprise.

Those two quick solutions alone will save you time, effort, and a flamewar (or a kick from IRC).

• • •

The second part of my editorial now goes out to all those established in the Java language. People of a nervous disposition may not like what I am about to say, but please hear me out. If you have a bookshelf full of Java books (assuming you personally own them), study them closely and pick out which titles you use on a regular basis. Set these titles to one side, then turn to the remaining books and ask yourself if you really need them. Here comes the painful bit: give them away to someone who can make better use of them; regardless of the age of the books, a beginner will make better use of them than you. I did tell you it would be painful.

All beginners need a hand to hold. I appreciate that, but it can quickly become an exercise in doing their work for them. If you've spent all that time learning, so should they. The harsh alternative is to send them an invoice for the work. I hate having to be cruel to be kind, as do most people, but the line has to be drawn somewhere. If you get a reputation of just handing out code when someone asks for it, then everyone will flock to you and you'll get bored of that situation very quickly.

Think about the title of this month's editorial (a term I heard from another guitarist, Don Potter). If you don't have something to hand down to the next generation of programmers, what will the future hold for Java in the marketplace and the industry as a whole?

jasonbell@sys-con.com

AUTHOR BIO

*Jason Bell is a programmer and senior IT manager for a B2B Web portal in York, England. He has been involved in numerous Web projects over the past five years, the last two of which have been servlet-based.*

## J2SE INDEX



# From COBOL or RPG to Java

## It's easier than you think

WRITTEN BY
DANTE CASTIGLIONE

**This article, tailored for seasoned COBOL and RPG programmers, provides a jump start into the Java programming language and the concepts behind it; it may also be useful for anyone who is stepping into the Java arena for the first time.**

Usually depicted as a language suitable only for those with a strong background in C or C++, in reality Java is a modern, eclectic language that builds on a variety of languages, including C and C++, and also borrows many features from Pascal, BASIC, and, yes, COBOL. In the last few years RPG has been moving closer to COBOL, so close in fact that many people say it's just COBOL written on an RPG specification sheet.

In the end, Java is just another language to learn and master. You did it with COBOL and/or RPG, you can do it again.

The difference is that with COBOL and RPG, the path to mastering them appears quite straightforward. You learn structured programming, some instructions, and then make your first attempt with a compiler. The more knowledgeable you become about a specific compiler, the more valuable you are as a programmer.

On the other hand, when facing Java programming, you get tangled in a jungle of unfamiliar terms and concepts: object-oriented programming, threads, sockets, HTML, browsers. And it looks like there's always more…and more.

That's just the beginning. In fact, you also need to learn about IP networks, but don't worry, it's not rocket science.

### In the Beginning: The Birth of Java

Maybe you've noticed that many gadgets and appliances in your home or office contain some software; for example, a microwave has a piece of software in it to manage the clock and turn it on and off. This is an entire industry called *embedded software*. One of the problems in this field is that microchips change, and change fast. So software must be rewritten and tailored to work with the new processors.

James Gosling and his team came up with a new idea: make a language that's portable and able to run in as many platforms (please read environments) as needed by simply modifying the compiler for the new chipset.

This was not the first attempt to create a portable language. Do you know which one was the first? COBOL. The acronym, as you probably know, stands for COmmon Business-Oriented Language. The key word here is "Common" as it was truly a standardization effort, aimed at the write-once, run-anywhere concept so dear to Java. The C language was the other important try. The two of them failed in the long run. Compilers diverged, and soon there was no portability at all. If you're a COBOL programmer, you surely know what I'm talking about. RPG has better portability, but it's limited to a few IBM proprietary operating systems, so it doesn't count when looking at the industry as a whole.

The Java idea was good; the timing was not. The project failed. A few years later, the Web was in full fashion and Sun had this product ready.

They thought it was a good idea to download a program from the server, along with a Web page and its graphics (if any), and run the downloaded program in the client machine, no matter which hardware/operating system/browser combination you were using. It was a good idea, but over time the language became more than that.

### Let's Stop Talking and Start Coding

The simplest example may be the classic one, Hello World. There's some good news: in Java, you need to write fewer lines than in COBOL or even RPG to get something done.

```
/*
 * this is a remark
*/

class HelloWorld {

  public static void main(String
args[ ]) {

  System.out.println("Hi, I'm your
first Java program. Please call me
class.");

  } // this is the end of the 'main'
method

} // this is the end of the class
```

Here you can see a couple of interesting things. First, a program in Java is called a *class* and routines are known as *methods*. They're almost the same; we'll talk about the differences later. In the code sample, the (one and only) method is called "main". This may remind you of the MAIN SECTION of a COBOL program (mandatory for some compilers). This method has a special "signature" (style of writing) that must be followed exactly if you want your program to be able to run from the command line; that's why you may find the second line a bit strange.

You'll also find that you can use the good old remarks in Java. In the code sample there's a multiline remark at the beginning (in fact, it can have one to *n* lines) that you may find familiar, as it looks like the mainframe or mini control language remarks, and even quite like

the COBOL or RPG remarks. The Java toolkit comes with a tool called Javadoc that can be used to extract the remarks into a group of HTML pages, so they can be used as a basis for documenting an application. Look also at the "//" character group; it marks the beginning of an inline remark (like the ones signaling the end of the method and the end of the class).

## Compiling

The tools you need for your first attempts at coding and running Java are free and easily downloadable from the official Web site, http://java.sun.com/j2se. Install them on your PC, Mac, or workstation. This is the best approach at first; once you feel comfortable with this environment, the next step is to try more sophisticated tools to develop and run an application. First you need to download the Java Development Kit (JDK) for your platform of choice (PC, Mac, Unix, Linux, etc.).

Once you've installed it, follow these instructions for a standard Windows PC (it's more or less the same for the other platforms):

1. Open a command (DOS) window.
2. Create a folder for your code (mkdir folder_name).
3. Type in the example using a text editor (it may be the Window's Notepad, WordPad, or other that you like).
4. Compile the example using the command javac (Java compile).

*Note:* Remember there are some rules for a Java source file:

1. The file must have the .java extension, so its name should look like Class_name.java.
2. class_name is in fact the name of the program, so it must be the same name you typed in the class signature, that is, when you declared the class typing "class Class_name".

## Testing

Once you complete the previous steps, you should have two files in your code folder: Class_name.java (containing your source code) and Class_name.class (holding your compiled code).

To run a Java class (program) use the following command:

```
java Class_name
```

This brings the Java runtime environment to the computer memory, and will in turn load your class (program) and execute it. This is the same thing you do when invoking a COBOL or RPG program from a control language script. After this, your Java class can load other classes and transfer control to them; again, this is quite similar to the concept of a run unit in COBOL or RPG. In Java, this is such a natural thing to do that it doesn't even have a name (i.e., there's no "run unit").

## Achieving Portability in Java

When you compile a Java program, the compiler first checks your source code for errors, then it translates the source into "Java bytecodes" or just "bytecodes" – machine-like instructions not targeted to any specific platform (operating system and hardware). During runtime, there's an interpreter that reads the bytecodes and causes the execution of instructions suitable for the target processor (microchip). Think of this as the runtime provided with some PC COBOL compilers (like RM-COBOL from Ryan-McFarland or the old Microsoft COBOL). The bytecodes are said to be instructions for a "virtual machine." Think of the virtual machine as a processor that's well defined, but that you won't find at a hardware store; it's implemented in software, on top of existing microprocessors.

You'll find two different packages on the Java Web site: the JDK that I men-

tioned before, and the Java Runtime Environment (JRE). The latter provides only the runtime, without the programmer's tools, and is useful for deploying applications.

## Data and Objects

In COBOL you use data by declaring variables. It's not so different in Java. As a matter of fact, it's better. Java is a modern language, so it builds on the strengths of its predecessors.

And there is something good: you can have basic data types as well as complex ones. Maybe you'll think that this is no news, that you actually use complex data types in COBOL by defining groups of data items in the working storage section. Oh, yes, I'm sure you do it, but here you can have executable code perpetually linked with each complex data type that you define.

This is not a minor thing. It brings up a whole new palette of resources and tricks and ways to do things. In fact, this new possibility is so cool it's called *object-oriented programming*.

An object is just that: a group of fields plus related code, all in a package that can't be (legally) opened. This may not be the best theoretical definition (and maybe it's not even a good one) but I can assure you, from my teaching experience, that it enables you to grasp the concept.

Listing 1 provides a simple program that:

1. Receives parameters from the command line
2. Translates the text-based parameters into numbers
3. Summarizes the numbers
4. Prints the total using the system console

Let's analyze it in more detail. First, look at the main() method signature. Inside the parentheses you'll find a data definition, "String args[ ]." This is an array of text parameters received from the command line. It's a mandatory thing for any Java class (program) that wants to be able to run from the command line. You must accept an unspecified number of text parameters from the command line or the class won't be able to run from it. Note that this program needs two parameters to run properly. It won't do any damage to pass more, but if you pass none, or just one, you'll get an error message.

After the method signature you'll find two data definitions. One is a simple, basic variable called a primitive data type in Java. Other object-oriented languages force you to treat absolutely everything as an object; that may be a good thing to do from a theoretical point of view, but in everyday, commercial, production programming practice, it's really a pain in the neck. There's no sense in using an object when all you need to do is count from 1 to 10. (You can use an object if you're a purist, or if you just want to; this is my opinion – shared by many people.)

The next one is the definition for using an object. Note that primitive datatype names are in lowercase, and class names must be written with an initial uppercase letter. Java is very case-sensitive, and that's one of the minor things you'll find annoying at first. But remember, COBOL was good because it was very precise and it required a lot of order from the programmer; Java shares the same philosophy, so I guess you'll learn to like it. Using code from other programmers is probably the most useful technique for a professional programmer; in an object-oriented language it's surely the most important technique!

After the definition, the object needs to be initialized. Do so by invoking a special method called a *constructor* – you can conceptually compare it to that

frequently used COBOL routine for opening the files; there's no exact example for RPG-loving programmers since we know that RPG opens the files automatically. However, if you're an RPG programmer who knows your compiler, you'll remember that there is one thing called *first-cycle processing* implicit in the RPG runtime cycle where files get opened.

So, we created a program and it's running in memory. We defined a simple data item and declared an object in memory. The Java environment went to disk, found the class definition, and instantiated an object into memory using that class as a model. I guess this is enough preparation; now let's do our work.

We transfer control from our object (program) to the other object (program) by invoking a method (subroutine) in it. That's what the line c = b.parseInt(args[0]) does. It calls a method called parseInt (short for parse integer) in object b. Object b is equal to the class Integer, so it has all the methods in this class.

I told you there were some differences between a method and a subroutine. It's time to outline them.

Subroutines are like methods in that you can transfer control to both of them. In COBOL or RPG, you can transfer control to any other subroutine in the same program, but if you transfer control to another program, you can't choose the subroutine. It will execute the whole program from the beginning. In Java, you're free. You can execute any method (subroutine) in your class (program) or any method in any other class (if you've previously loaded it into memory; anyway, there is an exception to this).

The other important difference is that you can't pass data to a subroutine; it will look for it in the program's data definitions. But you can pass data to a method; put the data (or a reference to it) between the parentheses following the method name. And a method can also give back data. Let's look at that strange line again:

```
c = b.parseInt(args[0])

<---              <-----

Output            Input
```

As you can see, the method receives the first occurrence in the array args[ ] as a parameter. That's the easy part. The one thing you'll find a bit weird is that the method invocation is to the right of an assignment expression (the "=" sign). In COBOL or RPG you must use a variable to do this. Write the subroutine so that it puts the output value in the variable, then move that variable to another. In Java, you can do that in one easy step. The output of the method parseInt is (as its name implies) an integer. And that integer goes to variable c, also an integer.

To end the program we invoke a method called println (meaning print line) in a special system object. This object doesn't need to be instantiated and initialized, because it's a *static method*. Conceptually this is not very different from writing to the system console in COBOL or RPG; just the code is different.

This is it: Java is easier than it looks at first glance. You can try it at home or at work on a PC; the same code will run on any platform, so you're leveraging your abilities. Java will program almost anything. Once you can code and hack the simple examples, the next best thing is to concentrate on learning one API (set of classes) that you find interesting, understandable, and profitable. Profitability depends on where you live and who your clients are; some will find that AWT (the windowing API) can get them their first real work in Java. I usually teach the servlet API, because it's kind of easy to find some work to do for online applications.

**Listing 1**

```
/*
   Numbers
   An example for CoBOL or RPG programmers
*/
class Numbers {
     public static void main(String args[ ]) {
          int c; // a simple integer variable for holding a number
          Integer b; // a complex data type, an Integer with associated
code
          b = new Integer(0);
          // a complex data type needs initialization
          c = b.parseInt(args[0]);
          // using a subroutine (method) in the program
          // "b" (an object) we translate a textual
         // command-line argument into a numeric value
          c = c + b.parseInt(args[1]);
         // the same for the second parameter
          System.out.println("The sum is " + c);
     } // end of method
} // end of class
```

Download the Code! www.JavaDevelopersJournal.com

## Resources

- ***The Java Tutorial:*** http://java.sun.com/docs/books/tutorial/
- ***New to Java Programming Center:*** http://developer.java.sun.com/developer/onlineTraining/new2java/
- ***The Java technology zone:*** www-105.ibm.com/developerworks/education.nsf/dw/java-onlinecourse-bytitle
- ***Java sites for mainframe and medium platforms:*** www.ibm.com/s390/java, www.ibm.com/as400/java
- ***The Java Boutique:*** http://javaboutique.internet.com/

**Author Bio**

*Dante Castiglione has written more than 30 articles for speciality magazines and newspapers in Latin America, and is the author of* Building Intranets. *He has taught at seminars and conferences in Argentina and participated in the IETF group defining geographic location and privacy standards.*

dantec@abastonet.com.ar

# Building Installers for OS X

written by Ian McFarland

## Make your Java apps available

Java development on OS X is similar to Java development on any platform, particularly any Unix platform. The differences are in how your code integrates with the platform. Java lacks a cohesive platform integration strategy, so running a Java application usually doesn't have the same feel as running a native one.

## "Java on OS X is a first-class citizen"

In contrast, Java on OS X is a first-class citizen. You can integrate your app so well that users probably won't even know they're using a Java application. You can package your app so it has one of those lovely 128 x 128 icons and can be launched with a double-click; it can even be bound to particular file types so it's launched when the documents are double-clicked. Swing apps also get the luscious Aqua user interface free, and with a couple of lines of code you can also tell the runtime to let your app use the system menu bar and the hardware acceleration that you won't find on any other platform.

OS X also comes bundled with a great set of developer tools, all at no cost. I'll be using one of them, the PackageMaker, in this article.

### Creating a Native Installer

This article shows how to package your Java application into a native OS X installer. I'll use one of the many free development tools that come with the platform, so to follow along, you should be running OS X and have the developer tools installed. Depending on when you bought your system, the developer tools installer may be on a separate CD or preloaded on your system under /Applications/Installers/Developer Tools/. If you can't find the installer, you can always download it from the Apple developer Web site. You'll have to join the Apple Developer Connection, but the minimum membership is free and provides access to the tools. (You should do this anyway if you plan on doing any amount of development for the Macintosh, as it provides access to lots of good information.)

If you haven't already done so, run the installer by double-clicking it; you'll need an administrator password to complete the install.

The installer we'll be creating looks like the one you used to install the developer tools. (They eat their own dog food at Apple, so we're using the same tool they used to create their installer.) Figure 1 shows the installer I built for Ant 1.5, and Figure 2 shows the icon. It all looks very official.

There are two important versions of the package installer. We're documenting version 1.0, which came with OS X 10.0 and 10.1; OS X 10.2, code named Jaguar, which shipped on August 24, comes with version 1.1. Although this tool is much nicer that 1.0, I've opted to cover 1.0 for one important reason: installers created with 1.1 may not be backward compatible with OS X 10.1 and earlier. (This article was written before the final release was available, so this may be addressed by the time they ship.) The extra pain of using 1.0 is minimal, so you may as well make your Java apps available to the widest possible installed base.

The installer we create contains three groups of files; we want to install these files on the user's machine, which PackageMaker packages into an archive file. There are scripts, such as preinstall and postinstall, which are executed at various points during the install process, and a set of files that contains documentation presented to the user when the installer is run. We'll look at each of these in turn.

The files to install are the application, whether a double-clickable JAR file or a fully packaged .app bundle, and any other files you may wish to distribute with your app. Good examples are documentation files, configuration files, external data files, and the like. We'll cover how to build a fully packaged, double-clickable .app bundle, complete with its own icon and everything else in a subsequent article. For now we'll assume that the app is a double-clickable JAR file. (Any JAR file that includes a Main-Class: attribute will automatically be launched when you double-click it in the Finder on OS X.)

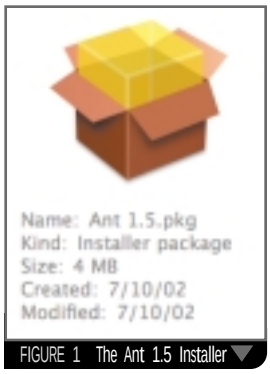


FIGURE 1 The Ant 1.5 Installer

You'll need to create a new directory that contains the files you want to install, with the files laid out the way you want in the directory the app is installed in. You'll typically put a folder named for your application in this directory as well as all the files you want to include. When you build the installer, you can specify whether the path contained here should be absolute or relative to a directory that the user chooses. For most apps, the latter is the right choice. Of course, for the Ant installer, I wanted to make sure that the files were installed in /usr/local/ant, and that symbolic links were created so that Ant would be included on the command-search path. Which one is appropriate depends on the kind of application you're working with, but if you can let the user have a choice about where the app lives, you should.

You'll then need to create a second directory for the installer resources at the same level as your first directory hierarchy. Be sure to name it something you'll be able to distinguish from the other directory when you run the installer. Place your scripts and documentation files into this directory.

The install scripts are scripts that are run before and after the archive containing the app is unpacked. There are several optional scripts you can include, each corresponding to a different phase of the install process. The scripts correspond to six phases: preflight, preinstall, preupgrade, postinstall, postupgrade, and postflight. This lets you check out the environment before the installer runs, move old files out of the way, do postinstall configuration, and clean up when you're done. The package installer runtime will execute each script at the appropriate time. They can be written in just about any scripting language, from AppleScript to PERL to a simple shell script. Many simple installers don't even need any scripts. Let's look at the preinstall script from the Ant 1.5 installer to see what it's all about.

```
#!/bin/sh

if [ -e /usr/local/ant ] ; then
    mv /usr/local/ant /usr/local/ant-pre-1.5
fi
```

My Ant installer preinstall script checks if there's a previously installed version of Ant, and if so, moves it out of the way before the installer unpacks the archive. It's nice not to clobber software someone may have installed by hand! The postinstaller then runs through, looking for any previously installed Ant extensions and copies them into its extensions directory. What you do in these scripts depends entirely on the app you're installing.

The scripts must be named according to naming conventions that the package runtime uses to determine which file to use when. In the 1.0 version of the package tool, the correct name for the preinstall script is <package-name>.pre_install, where <package-name> is replaced by the name of the installer you're creating (without the .pkg or .mpkg extension that all installers have). As of this writing, it looks like 1.1 will remove the underscore character. They do need to be named correctly in order for the package runtime engine to run them, so check the help files for the correct naming convention for the version you're using. They should have the read and execute bits turned on. (chmod a+rx *_install in your resources directory will do this.)

Once your scripts are written, you can add more files that describe the app so that the user is presented with information when installing it. There are three main files to be concerned with here: Welcome, ReadMe, and License. These can be in any of four file formats: HTML, RTF, RTFD, or plain text. They must have the appropriate extension for the file type: .html, .rtf, .rtfd, or .txt, respectively, and again must be named correctly in order for the installer runtime to know when to display them.

You may also include images in your HTML files, and they must also be included in the resources directory when you build your installer.

**"OS X also comes bundled with a great set of developer tools, all at no cost"**

These files are presented to the user in the order specified earlier. If you include a license file, the user will also be prompted as to whether he or she agrees to your license terms before being allowed to continue the installation process. These files can even be easily globalized, so the single installer will present information in the language of the current locale. You'll have to write the localized content yourself though.

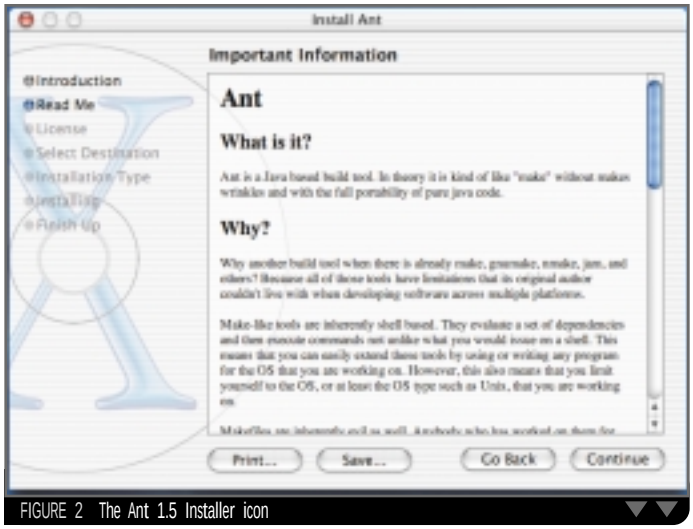


FIGURE 2 The Ant 1.5 Installer icon

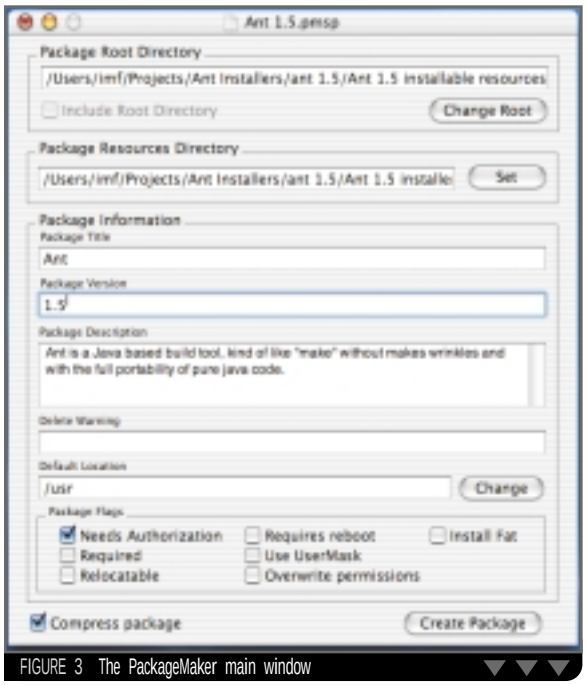


FIGURE 3 The PackageMaker main window

Last, you can even include a background image. This can be a .jpg, .tiff, .gif, .pict, .eps, or .pdf file. (Between Quicktime and Quartz, it must have been too easy to include so many.)

## Running PackageMaker

Once your files are all prepared, it's time to launch PackageMaker. You'll find it in /Developer/Applications/. It will present you with a single window (see Figure 3), but without any of the fields filled in.

In the Package Root Directory box, enter the full path to the first directory you created, the one that contains the actual files to install. You can either browse to it by clicking the Change Root button or enter the path if you know it.

In the Package Resources Directory box, specify the resources directory, the second directory you created.

In the Package Information box, fill out the various text fields: Package Title, Package Version, Package Description, with text that's appropriate for your app.

The Default Location field is important and works with the Relocatable checkbox. It specifies which path your app should be installed in.

If your installer is for a regular application, the default location should be /Applications and Relocatable should be checked.

However, if you need to install in a particular directory, specify it here. Remember, any subdirectories inside the Package Root Directory will be installed as subdirectories of the directory specified here. Most likely you'll also need to define that the specified path needs to be on the root volume. I'll show how to do that in a moment.

Depending on where your app needs to be installed and what your install scripts need to do, you may need administrator privileges to complete your install. Avoid requiring this if you can, but if you're installing in system directories or creating new users (as I do in my forthcoming Tomcat installer), you'll need your scripts to run as root for the install to work.

PackageMaker handles this for you. By checking the Needs Authorization box, you can instruct the package you create to request an administrator password. The installer then executes the scripts with the appropriate privileges. The Developer Tools installer worked this way.

**"Any JAR file that includes a Main-Class: attribute will automatically be launched when you double-click it in the Finder on OS X"**

The Required checkbox is only relevant if you're going to bundle this package into a Meta Package, a construct that allows users to select optional subpackages, as the Developer Tools package did. I'll cover how to do that another time.

The other checkboxes are fairly self-explanatory and can typically be left unchecked when installing a Java application. Try to avoid requiring a reboot. Now that Mac OS is

based on Unix, it can easily stay up for months at a time. Don't make your users sit through a reboot, unless it can't be avoided.

When you're happy with the settings, click the Create Package button and save out your installer.

The last little trick is the one I alluded to earlier. If your package needs to be installed on the root volume, you'll need to resort to using a text editor to tweak a file in the package you just created. (The 1.1 version has a checkbox for Root Volume Only.) Control-click or right-click on the package you just created and select Show Package Contents from the context menu that pops up. Then drill down to Contents /Resources/English.lproj to find the <package-name>.info file. This contains several of the settings we created using the graphical user interface. You can open it by dragging it onto TextEdit or your favorite text editor. You'll need to add a package flag:

```
rootVolumeOnly YES
```

Insert this line above the other package flags and save.

Now all you have to do to share your application with the world is to archive it and post it somewhere. The safest way to do this is with gnutar: gnutar -czvf <package-name>.tgz <package-name>.pkg. If you're distributing your app this way, make sure you don't have any filenames longer than 32 characters, since StuffIt Expander still doesn't maintain long filenames correctly, and, by default, browsers will open downloaded archives using it.

**"Now that Mac OS is based on Unix, it can easily stay up for months at a time"**


AUTHOR BIO

*Ian McFarland is the president of Neo Ventures Ltd., a software consulting company in San Francisco, and author of* Mastering Tomcat Development, *available in September from Wiley; he also maintains www.javaosx.com. He's been a Java developer since release 1.0 alpha 2.*

ian@neo.com



***–continued from page 7***


*The Java Community Process is the way the Java platform evolves. It's an open organization of international Java developers and licensees whose charter is to develop and revise Java technology specifications, reference implementations, and technology compatibility kits.*

Specifications don't solve business problems – they solve technology problems. Companies expect their programming teams to solve business problems. Take any average Fortune 500 programming team with little or no Java experience – a team with a deadline to meet. Show them the number of specifications being released with J2EE 1.4. When you multiply that number by the complexity, size, and learning curve of each specification, I bet they become scared and want to reconsider the use of Java. If Java can't continue to entice new programmers while retaining the programmers it already has, how will our community survive?

The original premise behind a J2EE container was glorious: abstract multithreading issues, server memory management, wire protocols, etc., from the programmers and allow them to focus on implementing solutions, not server infrastructure. Somewhere during this journey the JCP has shifted from its solution-oriented roots to merely implementing specifications. This trend must be reversed for the sake of our community!

At present, the open-source community appears to be the only one focused on building solutions, but signs of strain are clearly evident among even the most popular projects. As a case in point, consider the Apache Axis project. Axis is an innovative next-generation Web services engine that, when completed, will transform a number of interrelated Web services specifications into a solution that can stand strong as one of the cornerstones in any enterprise-scale project. The most disconcerting aspect of Axis is that since its inception in late 2000, there has yet to be a production-quality integer release issued. In my opinion, this is due largely to the deluge of specifications the project must digest. Each time I visit the project's Web site, I'm amazed at how the acronyms are multiplying like rabbits.

To the vendors and active members of each of the ongoing JSRs, hear your constituents speak! We need a break from this explosion of TLAs (three-letter acronyms). We aren't interested in expanding our already voluminous library of specifications – we can't get our arms around what we have today. As a community, we must pay attention to the beleaguered JCP process and realign it with creating solutions, like those routinely released by the Apache Software Foundation.

By taking steps now, we are investing in the future of both Java and the community that grew up around Java. The entire JCP process must thematically reflect our desire to build solutions that simplify complex technologies for programmers. In fact, the JCP process should continue to use the JSR acronym, but with new meaning: Java Solution Request.

JASON R. BRIGGS J2ME EDITOR

# Don't Believe Everything You Read

Where do market analysts get their figures? When you get a job as an industry or market analyst, do they give you a complimentary calculator that has a single button with a label marked "Random" on the front?

The reason I ask is that I read an article from Reuters a while back in which two analysts announced that there was an industrywide lack of applications and/or developers for smart phones.

So who wrote the 300+ applications currently hosted on midlet.org (a large number of which are admittedly games, but I certainly don't see a huge problem with that, at this relatively early stage)? I wonder if the commercial companies investing in games/app development for J2ME are aware that there is no one writing software for these phones? Perhaps they just have artists drawing little screenshots in a tiny 96x54 pixel box? Or maybe there's a single, rather lonely guy or gal out there being time-shared among all these projects? It's a scary thought because it would mean I'm writing to an audience of one.

In a rather interesting turnaround, while Microsoft may have decided they can do without Java on the desktop, they seem to have ascertained that there's a little too much mind-share in the J2ME world for them to completely ignore it – judging by a press release that arrived earlier this month. Insignia Solutions, after announcing a "strategic alliance" with Microsoft back in March (isn't that a bit like Cuba announcing a strategic alliance with the U.S.?), have announced the availability of their Mobile Foundation Software for Microsoft's Smartphone 2002. Mobile Foundation is apparently composed of three parts: a CLDC/MIDP environment, a provisioning toolkit, and an implementation of the Mobile Media API (JSR135).

While I have little interest in what Microsoft is doing in the mobile phone arena, I concede that the world is not one size fits all, and that some people might feel a certain amount of nostalgia and want to experience the blue screen of death on their phones, occasionally.

In all seriousness though, by all reports Microsoft has a rather enormous mountain to climb, considering their competitors, in what is for them a comparatively new market. For the Java world this news can be viewed in a fairly positive light. For J2ME to achieve the ubiquity that we, as developers, need in order to support a flourishing development marketplace, it *has* to be present on all devices, no matter what their market share. While having a consumer pop into a local mobile phone retailer to buy a new phone and ask for one that runs "those MIDP thingamajigies" is certainly a positive outcome from our point of view, a better result is the consumer who pops into a retailer and finds every phone is MIDP-enabled – without having to ask.

Take a global market of 500 million phones and assume all those phones were MIDP-enabled; if a new entrant to the mobile phone industry had only 0.1% market share, that's still 500,000 people. Five hundred thousand is not to be sniffed at, especially if that group is the most likely to purchase your newly developed application.

So, congratulations to Insignia for getting their product onto the MS platform.

• • •

In this month's ***JDJ***, Sean Campbell provides an introduction to Web services for devices using the kSoap library, and we also have the final installment in Bill Ray's series on remotely controlling your home's audio system.

jasonbriggs@sys-con.com

AUTHOR BIO

*Jason R. Briggs is a Java analyst programmer and – sometimes – architect. He's been officially developing in Java for almost four years, "unofficially for five."*

J2ME INDEX

# Mobile Web Services with kSOAP

## Develop complex SOAP Web services clients

WRITTEN BY
S.D. CAMPBELL

**The new kid on the Internet technology block is Web services and its implementation technology, SOAP. Simple Object Access Protocol is an XML vocabulary used to describe messaging and remote procedure calls between distributed components.**

The power of SOAP is its flexibility, which comes from SOAP's foundation in XML, and its ubiquity, which comes from the use of HTTP as a primary transport mechanism.

Any modern language that works with the Internet is likely to support both XML and HTTP, and Java is no exception. There are, in fact, many good toolkits for using SOAP with Java and Apache Axis is one of the most popular. Choices abound for the developer working with J2SE or J2EE.

### What About the J2ME Developer?

In the world of mobile devices, a world still constrained by hardware and network limitations, the concepts behind distributed computing are slowly taking root. Why cram all the functionality of a stock ticker or m-commerce application into one little MIDlet when you could offload much of the functionality to a server with more resources? Not to mention the flexibility it provides for system and database administrators. How often do they wish they could fix simple problems by logging in from a Palm Pilot or BlackBerry?

SOAP provides distributed functionality in a manner RMI and CORBA cannot. It's language agnostic, so a MIDlet can converse with a legacy COBOL application wrapped within a PHP SOAP service. It's also lightweight; there's no need for the resource-hungry ORB platform to support it. One small toolkit, kSOAP, weighing in at a whopping 41KB, is all that's needed to bring wireless Web services to your MIDlets.

### Application Architecture

The Web service I'll be looking at is a simple example of what can be accomplished. The taxCalc service takes two parameters, a tax rate and a subtotal, and calculates a total for a given purchase.

The service is written in PHP using the NuSOAP toolkit, a simple and easy way to prototype a Web service. In a production environment, once I had the definition of the service API frozen, I would then deploy a more robust version in a J2EE environment using Axis. For the purposes of this article, however, the use of PHP is handy to illustrate just how flexible and language-neutral SOAP can be.

The power of using a MIDlet is that it's deployable across a wide range of mobile devices, from a Motorola handset to a Palm Pilot or RIM BlackBerry. This provides the same SOAP client with a wide range of platforms for its deployment, and opens the door for a greater use of the Web service.

As can be seen in Figure 1, the architecture of our Web service is quite simple. On the server side we have an Apache Web server hosting the PHP service, which exports one interface function: taxCalc(). The endpoint for this service is available via www.whytewolf.ca/ws/taxCalc.php.

On the client side we have our J2ME client device running a MIDlet, which uses the kSOAP library to access the SOAP service. Messaging is done through the HTTP protocol, which is supported by all MIDP-compliant devices.

While you might think that a more complex Web service would require a more complex architecture, that's not necessarily the case. Any given application that uses Web services would only need to access a separate node for each service it consumes. If the same server or application provides multiple services – especially if multiple functions are exported by the same service – the application architecture is greatly simplified.

Ultimately, no matter which services are used or how many are available to the client, SOAP is about XML messages and the data they contain. It is these messages that the entire SOAP protocol is built upon.

### SOAP Messages

As stated earlier, SOAP messages are simply a specific XML vocabulary used to encapsulate document or procedure calls. As can be seen from Listing 1, the root element of any SOAP message is the envelope tag. Inside this envelope the request and parameters are transmitted to the server and the response is transmitted back.

Note the use of XML Schema data types to encode the data that's being sent as parameters. In Listing 1 the function call to taxCalc is made using a taxCalc element to enclose the two parameters, rate and sub. Each parameter element is given a type via the xsi:type attribute. The xsd:string type is an XML Schema string that, once the server receives it, is dynamically cast via PHP's internal mechanism to a numeric value.

The request is transmitted as an HTTP POST request and sends a special header: SOAPAction. This header indicates the specific service and function being called by including that information as a URI. In the case of taxCalc, what is used is a URN that indicates the service and function. This information is often useful, but the SOAP spec doesn't require that the server use it. The same URN is also used as a namespace URI for the service request element taxCalc.

Looking at the response envelope in

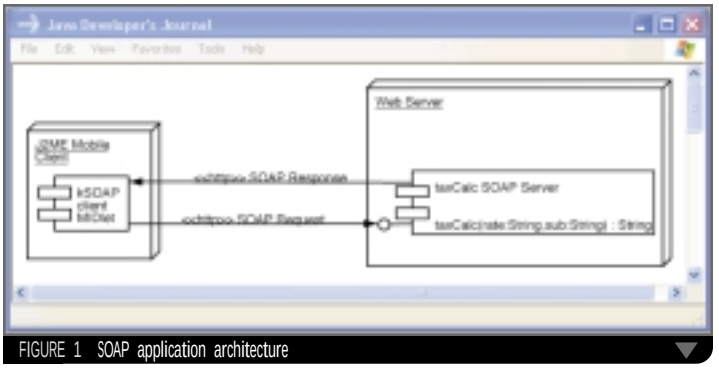



FIGURE 1 SOAP application architecture

Listing 2 we see that again the data being transmitted back to the client is wrapped within an envelope element's body tag and encoded using XSD data types. The response element taxCalcResponse wraps the actual value transmitted, in this case stored in a no-name element of type xsd:float, a floating-point number.

What if there's a problem? How does SOAP indicate a service failure and not leave the client hanging? The SOAP spec calls for the use of a Fault framework to indicate exceptional conditions in the execution of a service.

Listing 3 shows the anatomy of a SOAP fault. Like the request and response, a fault is transmitted using a SOAP envelope element, but instead of providing RPC calls or responses, the envelope's body contains the fault information. Four fault tags are transmitted: faultcode, faultstring, faultactor, and faultdetail. Faultcode and faultstring are mandatory. These two describe what type of fault occurred – if it was bad data from the client, a server failure, a version mismatch, or a miscommunication – and provide a message explaining the problem. As can be seen in Listing 3, the faultcode is client, meaning the client sent invalid data. The faultstring explains: taxCalc() can accept only nonzero subtotal values. The other two fault tags, faultactor and faultdetail, are optional and provide extended information on the cause of failure.

As can be seen in the examples, the SOAP 1.1 specification provides a highly detailed and flexible way to transmit data in both directions as well as indicate problems with the execution. However, with that flexibility comes a price: complexity. Parsing out the required details and marshaling the required data can be quite a task. Luckily for MIDlet developers, the kSOAP toolkit simplifies matters greatly.

For more information on SOAP, check out the Resources section at the end of this article.

## The kSOAP Toolkit

The kSOAP project is based on kXML, a lightweight pull parser designed specifically for use with MIDP. Enhydra, a provider of J2ME and J2EE solutions, hosts both kSOAP and kXML. Both toolkits are available through the Enhydra Public License, and come packaged in a single JAR for use in a SOAP environment – handy if the MIDlet needs to parse additional, non-SOAP–related XML.

One of kSOAP's biggest strengths is its relative simplicity. Most enterprise-level SOAP toolkits often rely on the use of a Web Services Description Language (WSDL) generated proxy object to make function calls. This is useful if a service has a published description and when the toolkit requires the instantiation of several different client and transport objects. While kSOAP doesn't support WSDL, it does make calling a service relatively painless. Only two objects are required: the SOAPObject and Http-Transport.

In addition, kSOAP makes it very easy to capture fault data. The toolkit maps all SOAP faults to an exception object known as SoapFault. In this manner, a Fault can be caught and handled like any other exception in the MIDlet.

The development environment used for this article is Sun's J2ME Wireless Toolkit. To build an application using kSOAP, first download the ksoap-midp.jar file from ksoap.enhydra.org. Place this JAR in the application's lib/ directory. The WTK will add all JARs in lib/ to the classpath.

Import all the necessary classes by inserting the following statements:

```
import org.ksoap.*;
import org.ksoap.transport.*;
```

Now your MIDlet is ready to use kSOAP.

## The taxCalcClient MIDlet

Up to now I've been laying the foundation for developing the taxCalcClient MIDlet. Now I'll dissect the construction of this application. While only specific relevant portions of the source code will be explored, you can download the complete application with source code from the ***JDJ*** Web site, www.sys-con.com/java/sourcec.cfm.

As shown in Figure 2, the first screen the user interacts with is a TextBox requesting the amount of the purchase. This is the subtotal value that will be passed to the taxCalc() function on the SOAP server. Looking at the MIDlet source we see the private variable rate has been set to +7%. This is the tax percentage that will be passed to the taxCalc() function as the rate parameter.

The Calculate event is fired when the right soft key is pressed. This calls the getTax() function (see Listing 4), which provides MIDlet's SOAP functionality. The heart of this function is:

```
try {
SoapObject client = new
  SoapObject("urn:soap-whytewolf-ca:
  taxcalc","taxCalc");
client.addProperty("rate",rate);
client.addProperty("sub",amount);
HttpTransport ht = new
  HttpTransport("http://www.whytewolf
  .ca/ws/taxCalc.php",
"urn:soap-whytewolf-
ca:taxcalc#taxCalc");

taxMsg.setText("$" + ht.call(client));
            }
```

To prepare our client to use a SOAP server we create a new SoapObject, passing the constructor the Namespace URI for the SOAP call and the name of the function being called. This SoapObject also needs to be prepared with the parameters the function accepts. Recall our taxCalc function took two strings: rate and sub. Each of these properties and the values to be passed through them are added to the client

FIGURE 2
MIDlet input screen

FIGURE 3
SOAP response screen

FIGURE 4
SOAP fault screen

object using the addProperty method, which takes the name of the property and the value of the property as parameters.

While these are provided as name-value pairs, some SOAP servers don't check the property name and instead use the values in the order they were passed. For this reason I suggest adding properties in the order the function would expect the parameters, in this case, rate first and then sub (see Figure 1 for the function's signature).

We now create a new HttpTransport object, which will provide the needed functionality to actually call the SOAP service. We pass the constructor the endpoint URL for the service and the SOAPAction URI we'll be calling the service with. To execute the service, call it using the call() method, passing call() the SoapObject that will invoke the service. Call() returns whatever value is returned from the SOAP service, in this case, a float that we cast to a string and use in the StringItem of our second screen to display the results of the tax calculation. Figure 3 shows the resulting screen, assuming all goes well.

As has already been indicated, the taxCalc SOAP service will not accept negative or zero-valued rates or subtotals. Should a client send anything other than a nonzero value for the rate or subparameters, the service will return a SOAP fault.

The first catch clause in getTax() will catch any SOAP fault exception the call() method throws. As seen in the following code, retrieving the SOAP fault information is quite simple:

```
catch (SoapFault sf){

taxMsg.setLabel("FAULT:\n");
   String faultString = "Code: " +
sf.faultcode + "\nString: " +
sf.faultstring;

taxMsg.setText(faultString);

            }
```

The SOAP fault is stored inside an exception object known as SoapFault (quite a convenient name). Any of the four fault fields can be retrieved through their named properties, sf.faultcode, sf.faultdetail, sf.faultactor, or sf.fault string. As shown in the example, it's quite easy to output these into the StringItem to indicate to the user a fault has occurred (see Figure 4).

Having handled the fault, the MIDlet allows the user to go back to the first screen and correct the data.

## Conclusion

SOAP can be a very complex realm to explore, especially the XML mechanisms used to transfer data between disparate systems, languages, and toolkits. The wireless world of J2ME and MIDlets requires tools of small stature and great power. Luckily when it comes to SOAP, the kSOAP toolkit provides not only small size and great functionality, but also relative simplicity and ease of use for the developer. Using kSOAP, a J2ME developer can develop complex SOAP Web services clients in a remarkably short time.

## Resources


- *SOAP 1.1W3C Note:* www.w3.org/TR/SOAP
- *kSOAP toolkit:* http://ksoap.enhydra.org
- *Java 2 Platform, Micro Edition:* http://java.sun.com/products/j2me
- *Mobile Information Device Profile:* http://java.sun.com/products/midp
- *Sun J2ME Wireless Toolkit:* http://java.sun.com/products/j2mewtoolkit
- Gupta, S. "Combining RMI with SOAP." *Java Developer's Journal*. SYS-CON Media. Vol. 7, issue 6.
- *NuSOAP PHP SOAP toolkit:* http://dietrich.ganx4.com/nusoap/index.php
- *Apache Axis:* http://xml.apache.org/axis



### Author Bio

*S.D. Campbell is a senior consultant with ContractBank (www.contractbank.com/CWIC) and heads the Web development program at Applied Multimedia Training Centers in Calgary, Alberta. His consulting company, Whyte.Wolf, helps small businesses move into wireless and Web services.*

scampbell@whytewolf.ca

**Listing 1: SOAP request**

```
POST /ws/taxCalc.php HTTP/1.1
SOAPAction: urn:soap-whytewolf-ca:taxcalc#taxCalc
Content-Type: text/xml
Content-Length: 557
User-Agent: kSOAP/1.0
Host: www.whytewolf.ca


<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body SOAP ENV:encodingStyle="http:
   //schemas.xmlsoap.org/soap/encoding/">
 <taxCalc xmlns="urn:soap-whytewolf-ca:taxcalc" id="o0" SOAP-
ENC:root="1">
   <rate xmlns="" xsi:type="xsd:string">7</rate>
   <sub xmlns="" xsi:type="xsd:string">856</sub>
  </taxCalc>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Listing 2: SOAP response**

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2002 01:31:10 GMT
Server: Apache/1.3.14 (Unix) mod_perl/1.24 PHP/4.0.6
FrontPage/4.0.4.3 mod_ssl/2.7.1 OpenSSL/0.9.6
X-Powered-By: PHP/4.0.6
Status: 200 OK
Connection: Close
Content-Length: 510
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0"?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<taxCalcResponse>
<noname xsi:type="xsd:float">915.92</noname>
</taxCalcResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Listing 3: SOAP fault**

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2002 01:32:12 GMT
Server: Apache/1.3.14 (Unix) mod_perl/1.24 PHP/4.0.6
FrontPage/4.0.4.3 mod_ssl/2.7.1 OpenSSL/0.9.6
X-Powered-By: PHP/4.0.6
Status: 500 Internal Server Error
Connection: Close
Content-Length: 607
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0"?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcode>Client</faultcode>
<faultactor></faultactor>
<faultstring>Must supply a non-zero subtotal.</faultstring>
<faultdetail></faultdetail>
</SOAP-ENV:Fault>
          </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Listing 4: getTax() function**

```
private void getTax() {
            amount = input.getString();
            try{
                 SoapObject client = new
SoapObject("urn:soap-whytewolf-ca:taxcalc","taxCalc");
                 client.addProperty("rate",rate);
                 client.addProperty("sub",amount);

                 HttpTransport ht = new
HttpTransport("http://www.whytewolf.ca/ws/taxCalc.php",

"urn:soap-whytewolf-ca:taxcalc#taxCalc");

                 taxMsg.setText("$" + ht.call(client));
             }
             catch (SoapFault sf){
                 taxMsg.setLabel("FAULT:\n");
                 String faultString = "Code: " +
sf.faultcode + "\nString: " + sf.faultstring;
                 taxMsg.setText(faultString);

             } catch (Exception e) {
                 e.printStackTrace();
                 taxMsg.setLabel("ERROR:\n");
                 taxMsg.setText(e.toString());
             }
       getForm.addCommand(backCommand);
       getForm.setCommandListener(this);
       display.setCurrent(getForm);

   }
```

Download the Code! www.JavaDevelopersJournal.com


Parts 1 and 2 of this series (*JDJ*, Vol. 7, issues 6 and 9) demonstrated how I developed a remote control MP3 player by using a Bluetooth connection from my handheld (a Compaq iPaq) and employing a client and a server written in Java.

Tracks were weighted to increase (or decrease) their chances of playing, and each track had a preferred track that followed, so the system would generally play two or three tracks from a particular album before moving on to the next one. The system works pretty well, even though a recent ripping session increased the tracks available to around 3,000 – remotely controlling my listening is great, but not perfect.

Like many people, I have speakers dotted around my house, with two amplifiers (one upstairs, another downstairs) and switches everywhere to control where the sound actually comes out. This is fine, but remembering which switch does what is a pain. If I'm in bed I don't want to have to wander into the office to switch on the speakers in the bedroom, so it's time to add some speaker control to my setup.

In this article I'll show how a Java application can easily take control of physical systems with the right hardware, both controlling hardware and responding to real-world events (well, a doorbell), and, as this is the final installment, I'll also be talking a little about why I developed this application and where it will go next. The source code can be downloaded from www.sys-con.com/java/sourcec.cfm.

## Controlling the Speakers

First I had to sort out how I was actually going to control my speakers from the computer. I hung up my soldering iron years ago, so I wasn't about to start building my own circuits – luckily a company called Quasar already makes the perfect circuit for my needs. They actually sell several kits that would work, but I went for a serial-controlled box with eight relays and four inputs. I also didn't fancy putting the thing together myself, so I ordered it preassembled, just plug and go. Testing via Windows HyperTerminal is easy and quick, as the commands sent to the box are simple indeed (N1 turns on relay one; F1 turns off relay one).

Originally, I wired each relay to the same 9-volt supply that provides power to the box, using that voltage to switch two dual-poll/dual-throw relays, one for each speaker. With this arrangement, one relay on the box controlled one set of speakers, but also required a lot of wires and didn't work satisfactorily. So I simplified it by wiring each speaker through a relay on the box, one relay for each speaker, using up six of my eight available relays (for three sets of speakers).

Audiophile friends have pointed out that I should really be smoothing the output to avoid the coil in my speakers pulling a high load on connection (which might cause "popping" when the speakers are connected), and Quasar does sell a kit to handle this. However, the speakers in the bedroom are nice NXT flat speakers without a coil, and the other sets are cheap, so I'll see how it goes. Pretty quickly (once I had given up my extra relays) I had speakers I could switch on and off from HyperTerminal.

The relays in the control box will actually handle the mains voltage, opening up all sorts of interesting ideas. Controlling the doors and lights has been vetoed by the house's other resident on the grounds that a crashed computer could plunge the house into darkness, but there's certainly room for a few more boxes in the future.

Then it was just a matter of writing a server application that would listen to one serial port for incoming commands over the Bluetooth connection and on receipt send them to the relay control box. I first developed the output side, and stuck a GUI on the application so I could control the speakers from the server screen.

One of the great things about object-oriented development is this ability to add and remove components easily. While the development could have been tested using command-line arguments, it was quicker and easier to stick a user interface on it with a couple of buttons that could be removed later. During testing I discovered that for some reason it would switch only one speaker in each pair, despite the fact that it was sending the right commands. It was quite a long time before I finally realized that the relay control box just couldn't respond to the commands fast enough. Putting in a

Thread.sleep() command sorted the problem, but had the side effect of turning on a pair of speakers one at a time, which can sound a little odd. I guess I'll be digging out those extra relays at some point.

As mentioned in Part 2, the Bluetooth stack not only allows serial connectivity, but also maps two standard serial ports to the Bluetooth network. Using the TDK USB Bluetooth on the server means that COM4 represents an incoming Bluetooth connection, so this is what we'll be listening to. As I had already created a "SerialConnection" class to communicate with the relay control box (using COM1), it made sense to just instantiate another instance of that class to handle the Bluetooth side of the process. I then discovered to my horror that my application had stopped working completely.

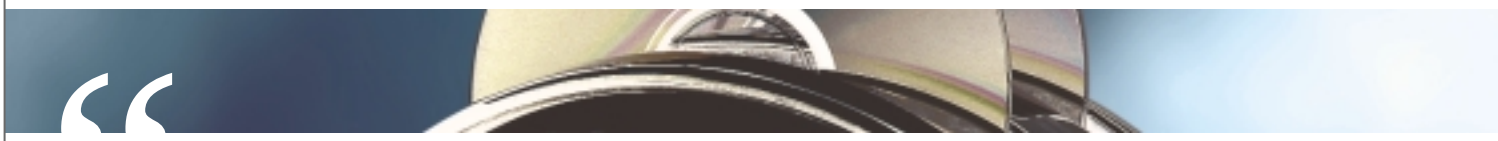
The problem was eventually narrowed to the opening of the serial port, and though I had not made any changes to the code, it was now apparently finding four separate ports, all called COM1! The only thing I had done since it worked was add the Bluetooth hardware, so for some reason that was causing the problem. It turned out that I was only checking if the name of the port found started with "COM1", and the Bluetooth drivers had added COM11, COM12, and COM13, all of which were considered fair game by my application.

```
    isBluetoothLink = true;
}
```

The Boolean "isBluetoothLink" is then used as a marker, so only the Bluetooth connection responds to messages received (see Listing 1). The strange arrangement whereby the office is on switches one and seven is due to physical difficulties with the lengths of wire available.

That got the server working, so I could start on the client. I was very keen on having a graphical interface, so knocked up a floor plan of the upstairs of my house (see Figure 1) so I could tap on a room to switch the speakers; it was a matter of putting the graphic on the screen and having it respond to taps in the right place.

Of course, PersonalJava doesn't support serial connections so I used James Nord's excellent implementation, which provides all the important bits of the Java Communications API for the Pocket PC. The outgoing serial port on the iPaq is COM8, and, as previously discussed, if COM8 is opened without a connection, the Bluetooth stack will reestablish a connection to the last place it was connected to. So I manually connected to my server one time, and can now run the application fine (until I use the serial port to connect somewhere else).



"I was very keen on having a graphical interface, so knocked up a floor plan of the upstairs of my house so I could tap on a room to switch the speakers"

This kind of problem is common enough when applications are moved between systems and should be guarded against; loops searching for things should stop when they've found them!

Once that was fixed, I instantiated two instances of my SerialConnection object and started testing, opening a serial connection from another PC with Bluetooth, again using HyperTerminal for testing. I decided on a very basic protocol with no perceptible handshaking at all. The client simply sent the name of the room followed by "on" or "off" and the server would switch the speakers.

Some sort of response from the server would have been better, and should probably be added for robustness, but for the moment I'm relying on any transmission problems being reported by the serial connection. Initial testing went well, except that the switching commands were being sent back to the PC making the request instead of to the relay-switching box. Again my lack of architecture design let me down, and what I had were two instances of my SerialConnection class communicating with devices with no way to communicate with each other. So I overloaded the constructor with one SerialConnection instance that would take the other as a parameter to be used to report when requests were made:

```
SerialConnection(CommPortIdentifier portId,
SerialConnection localCom) {
    this(portId);
    outputConnection = localCom;
```

## Testing the Application

Testing was remarkably smooth, with only one issue – the application could only be run once and the iPaq had to be reset before it could be used again. Obviously something was not being closed properly, and the serial ports were prime suspects; sure enough, explicitly closing the port sorted out that problem.

It would be nice if the application polled the relay box to get the settings so the display could reflect the sound coming out when it was launched, but I can normally hear if the speakers are on, so it's not a priority. The display isn't double buffered so there is some flicker, but that's not important in an application like this. The connection is fast though, taking a few seconds, and it's nice to be able to wander around the house, switching the music to follow.

The speaker control program is called "Conductor", but is still a separate application from the MP3 player. Since I want them both to respond to Bluetooth connections and run on the same machine, I combined them into a single application. This turned out to be remarkably simple, both on the server and the client sides. The MP3 player class simply instantiates an instance of Conductor and passes it any unrecognized commands received over the network, while Conductor does the same with commands received over Bluetooth, passing them on to the MP3 player. This also meant that Conductor would work over a network connection, making it much easier to combine the client applications.

The parent class of the Conductor client was just changed from a frame to a dialog (as only one frame can be used per

FIGURE 1 GUI floor plan

application in J2ME) and instantiated from the MP3 client. I also took the opportunity to move some buttons around on the MP3 client that were vanishing under the Jeode control bar on the iPaq. Now I had one client and one server talking over Bluetooth or network connections (specified on the client command line).

## The Doorbell Saga

We moved into our current house about a year ago and have never had a doorbell. We always felt that it should be something silly, but never saw anything that suited us, so this seemed the ideal time to look at the inputs on the relay control box. Since they can detect incoming voltage, I thought it might be interesting to connect a doorbell that could then control the sound playback. A cheap wireless doorbell turned out to be only a tenner and was remarkably easy to wire in; I just disconnected the speaker and ran the wires into the sensor on the relay control box. By polling the box with "I1", it will return "0" if nothing is happening and "1" if the doorbell is trying to chime. Since the chime lasts a couple of seconds there's no hurry in polling.

One architectural issue that came back to hurt me was the fact that I had always used event-driven serial communications. What this means is that the client makes a request of the server, but doesn't wait for a response. When the server sends its response, the client reacts to it without knowing that it was responsible for sending the request. This makes for quick code development, and if the responses from the server can be easily identified, then it's a fair way of working.

Now I was communicating with the relay control box, which has only very basic responses and it's much harder to identify what it's responding to. For that reason I needed to block until a response was received from the control box, so I had to change the event mechanism. It just stores the incoming message in a string and creates a method that checks the string and blocks until a message is received (see Listing 2).

The getMessage method also ignores messages starting with "#", as these are echoes of what was sent.

Polling the input in another thread is very easy, and when a "1" is discovered, the music is stopped, all the speakers in the house are switched on, and a predetermined MP3 file is played; then the speakers are switched back to their previous state and the music restarts if it was paused earlier. Listing 3 shows the polling loop.

Note that only one thread can talk to the relay control box at a time, so a Boolean is set if the speakers are being turned on or off (remember that due to the speed of the relay control box this will take half a second) and a poll might be missed. This works pretty well, though I did have to introduce another Boolean to stop the speakers from trying to be controlled while this thread is polling, and my head started to hurt.

The doorbell will play the file "door.mp3" when the doorbell is pressed, and for the moment this is the sound of an old-fashioned knocker banging against wood, but there is still an issue with range. Wireless technology is not nearly as common in the UK as in the States; basically, due to the limited amount of space available, the frequencies we can use are much more limited and transmissions are lower power.

The wireless doorbell I had bought claimed to have a range of 30 meters, which should easily encompass my house (Bluetooth works just about house-wide, and it's only supposed to be 10 meters) but I was horrified to discover that my 10 pounds worth of wireless doorbell actually has a range of about 2 meters. To ring the doorbell you have to invite visitors in and show them halfway up the stairs, where they can press the bell and be impressed. While this amused my wife a great deal, it's not really what I was after, so another doorbell was obtained, this time for 40 quid, which claimed to have a 100-meter range. I don't know where these figures come from, but this new doorbell actually has a range of about 4 meters, at best (admittedly, we are in the middle of a very humid heat wave at the moment, but from 100 meters down to 4 is still very poor). So some wiring was required, and now the doorbell is half wireless (as far as the bedroom) and half wired (into the office) and works properly. Relying on a server running Windows to provide a doorbell does worry me, but if we don't respond people will probably just bang on the door anyway so it's still not mission-critical code.

Of course, this left me with another wireless doorbell with an, almost, useless range, but easy to integrate with the rest of the system. Thinking it through, the place where control was most needed was the bathroom, where you might be lounging in the bath and decide some musical accompaniment would be good. Taking an iPaq into the bath is something I have done, but it will end in tears, so a cheaper alternative would be good. So the original cheap doorbell was wired into the second input on the control box, and the code expanded to poll it too and toggle the bathroom speakers when it is pressed. I'm rather taken with this idea, and may well buy another switch or two to control other functions.

## Conclusion

Now I can play music to suit my mood in the rooms of my choice, though the whole application is not yet perfect. The MP3 player leaks memory somewhere, causing it to crash after about 30 hours, and it still has an obsession with "Mr. President" by 4 Non Blonds for reasons I've yet to establish, but it is a program I use almost every day and I expect it to be available whenever I want it.

The process of developing this application has also been interesting, personally, as a reminder of why I got interested in computer programming in the first place. While many people now decide to study software development as a career choice, I got interested in programming because computers weren't doing the things I wanted them to.

In these days of supercomplex software and massively available shareware, it's easy to find a program that will do pretty much what you need without going through the effort of writing it. I admit that before this project I hadn't written any software for myself for years, just accepting the limitations of what was available. But writing software for fun is actually better than I remembered. If you don't have to worry about deadlines or insane functional requirements from inept management, or don't have to document everything, it's possible to concentrate on getting things the way you want and to deal with the intellectual challenge of solving the problems, which is fun.

Next I want to spread my control to the downstairs, but laying even more cable is something I want to avoid (the house is already looking like a slab of Swiss cheese). Some sort of Bluetooth link to another machine downstairs might be the solution, perhaps a Palm Pilot, which could then control the speakers downstairs. It also occurs to me that there's a WAP

profile in the Bluetooth specification that would provide the perfect interface for this kind of project and should then work with any WAP device. It would certainly be an interesting project. Fixing memory leaks and dealing with minor bugs is something I'll save for the software I'm paid to write – unless, of course, I get really bored.

### Useful URLs

- www.quasarelectronics.com
- www.bluetooth.com

bill@network23.co.uk

### Author Bio

*Bill Ray has worked for several telecommunications companies around Europe, including Swisscom, where he was responsible for the development of their Java-compatible DTV platform. He is security editor for* Wireless Business & Technology *and coauthor of* Professional Mobile Java Development, *published by Wrox Press*

**Listing 1**

```
public void messageReceived(String s) {
  if (!isBluetoothLink) {
    return; //Ignore input from the box, it's very dull!
     }
  System.out.println("Message :" + s);
  int switch1, switch2;
  String upperCaseVersion = s.toUpperCase();
  if (upperCaseVersion.startsWith("BEDROOM")) {
    switch1 = 2;
    switch2 = 4;
  } else if (upperCaseVersion.startsWith("BATHROOM")) {
    switch1 = 5;
    switch2 = 6;
  } else {
    switch1 = 1;
    switch2 = 7;
  }
  if (upperCaseVersion.indexOf("ON") > 0) {
    outputConnection.switchOnDevice(switch1, switch2);
  } else {
    outputConnection.switchOffDevice(switch1, switch2);
  }
}
```

**Listing 2**

```
public void messageReceived(String s) {
  nextMessage = s;
  return;
}

public String getMessage() {
  String tempMessage;
  while (nextMessage.equals("") || (nextMessage.charAt(1) == '#'))
{
    try {
      Thread.sleep(50);
    } catch (InterruptedException e) {
      System.out.println("Insomnia");
    }
    tempMessage = nextMessage;
    nextMessage = "";
    return tempMessage;
  }
}
```

**Listing 3**

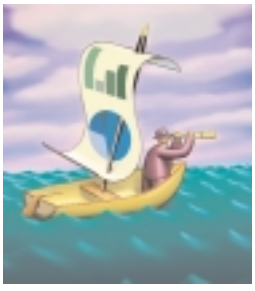
```
public void run() {
  while (notDead) {
    if (!tryingToChangeSpeakers) {
      isDoorbellPolling = true;
      actualSendMessage("I1\r\n");
      if (getMessage().endsWith("0")) {
        doorbellPressed = false;
      } else {
        if (!doorbellPressed) {
          doorbellPressed = true;
          soundDoorbell();
        }
      }
    } else {
      System.out.println("Missed Poll, speaker control in
         progress");
    }
    isDoorbellPolling = false;
    try {
      Thread.sleep(300);
    } catch (InterruptedException e) {
      System.out.println("Insomina!");
    }
  }
}
```

Download the Code! www.JavaDevelopersJournal.com

# Chart Your Course to Success in IT...


## ...order your copy of Java Trends: 2003

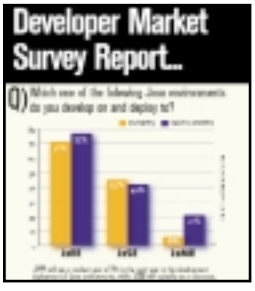
Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?



**EnginData Research** has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.

**EnginData**'s research is invaluable to leading IT vendors, integrators, Fortune 500 companies, trade-show organizers, publishers, and e-business organizations worldwide.




**Here's just a sample of the critical data points the Java report delivers...**

- ✔ Current and projected usage of languages and platforms
- ✔ Multiple rankings of hundreds of software vendors and tools
- ✔ Types of applications being developed
- ✔ Databases being used
- ✔ Purchasing patterns
- ✔ Company size
- ✔ Development and purchasing timelines
- ✔ Perceptions and usage of Web services
- ✔ Preferred Java IDE
- ✔ J2EE, J2ME, J2SE usage comparisons

As an IT specialist, **EnginData Research** focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- ✔ Time frame for developing Web services – enabled apps
- ✔ Percentage of apps with Web services today
- ✔ Sourcing and hosting Web services
- ✔ Perceived leaders in Web services tools and solutions

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of **EnginData Research**. Order your copy of the 2002–2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.

engindata RESEARCH

**www.engindata.com**

# HIP 1.0 PJ and J2

by Our World Live

REVIEWED BY MATTHEW DODD matthew.dodd@mausik.com

**Our World Live, Inc.**
5901 Warner Ave. Suite 21
Huntington Beach, CA 92649
**Phone:** 714 840-5520
**E-Mail:** info@ourworldlive.com
**Web:** www.ourworldlive.com

**Specifications**
**HIP PJ:** Any platform with PersonalJava 1.2 or JDK 1.1.8 support
**HIP J2:** Any platform with Java 2 support
**Pricing:** Development license starts at $1,695, deployment license starts at $20, and the ShareWare deployment license is $1.

**Test Platforms**
Sony VAIO FX150, 596MHz Intel Pentium III Processor, 20GB Disk, 128MB RAM, Windows XP v 2002
Compaq iPaq, 64MB RAM, PocketPC 2000

Interacting with software can be extremely annoying or incredibly exciting – depending on the user interface. While one GUI can leave you hopelessly confused and deny the execution of simple tasks because you can't find the right button, another GUI can draw you in with its appealing and conclusive features, making the implementation of even complex procedures a walk in the park.

For a long time, the development of graphically advanced user interfaces was something only desktop software developers could do, as Java simply lacked the respective toolsets. AWT and later Swing have tried to fill the gap, but are still greatly limited compared to Microsoft's Windows OS or Apple's Mac OS X.

With the growing range of end-user devices that directly support Java, it's increasingly becoming the one true platform-independent environment every developer has been waiting for – if only the graphical user interface frameworks could compete with their native counterparts on Windows or Mac OS.

This is where Our World Live's (OWL) Human Interface Package (HIP) comes in. HIP 1.0 for PersonalJava and HIP 1.0 for Java 2 (J2SE and J2EE) are two user-interface frameworks that provide a complete set of out-of-the-box APIs and, according to OWL, "transcend the limitations of AWT and Swing and revolutionize the development of Java GUIs in 100% Pure Java."

## HIP 1.0 PJ

HIP PJ, written entirely in Java, is OWL's user-interface framework that complies with PersonalJava 1.2 (JDK 1.1.8), which you'll find in most TV set-top boxes, Pocket PCs, and Nokia and Ericsson cellphone communicators. The most common virtual machine for Pocket PCs is the Jeode from Insignia Solutions, which I used on my Compaq iPaq for this review. Developers can use HIP PJ to implement a new and unique user interface for their Java applications (or applets). Each UI component can be arbitrarily shaped, with the mouse reacting correctly on it, and has a transparency property from 0 = opaque to 100 = fully transparent. Components can overlap; they can be dragged, and their visual appearance, including transparency, is automatically rendered accordingly. Skins can be defined, changed, and switched on the fly, making it easier to give applications the look you want. More than 80 classes, such as OWLButton, OWLDialog, OWLSlider, and OWLWindow, make up the powerful and flexible framework. The API is built and modeled after the AWT concepts and is easy to use.


## HIP 1.0 J2

HIP 1.0 J2 is OWL's user interface framework that's compliant with the Java 2 specifications. It's written entirely in Java but built along the Swing technology that's included in Java 2. It brings the same feature set to the Java 2 platform as its PersonalJava counterpart, HIP 1.0 PJ. UI components can be transparent, draggable, and have true free-form shapes with the mouse reacting exactly to the defined shape. As with HIP PJ, skins can be defined, changed, and switched at runtime. The framework consists of over 60 classes, such as OWLJButton, OWLJMenuBar, OWLJSlider, and OWLJTextArea, that can be used to develop state-of-the-art user interfaces

## Installing and Using HIP

Both HIP packages can be downloaded from OWL's Web site. HIP comes packaged as a JAR archive, thus it can be easily linked into any Java project. No additional configuration is necessary.

When a compiled project is run the first time, HIP asks for a license key and creates a license file that has to be stored beside the HIP JAR or in the system classpath. As long as HIP finds the license file in one of those two places, no further haggling with licenses is necessary. The license key mechanism is network-aware and will detect if the same key is used twice.

The included Read Me provides some basic information and pointers to the extensive API documentation on OWL's Web site. The documentation lives up to its task and provides a fully indexed and cross-linked

# bea

www.bea.com

Javadoc manual. If you have experience with AWT or Swing, soon you'll be using HIP like an expert. Starting with HIP, you'll notice that replacing Swing classes with their HIP counterparts usually works, speeding up your development time.

It would have been easier if OWL had included a default skin and some examples on how to use it to develop your own.

## Sample Program

Let's take a simple example and do some programming. The goal is to create a free-form crosshair that floats over the background (see Figure 1).

The task is simple. All you need are two graphics (JPEG or GIF), the picture of the crosshair (see Figure 2), a simple mask (see Figure 3), and some code for instantiation. Convenient multiparameter constructors or single methods for each representation can be used in both HIP frameworks. An example of the crosshair instantiation:

```
// HIP PJ code
crosshair = new OWLButton(crosshairBackground, null, null,
  null, null, crosshairMask);

// HIP J2 code
crosshair = new OWLJButton();crosshair.setImage
(crosshairBackground);
crosshair.setMask (crosshairMask);
```

This is much easier than creating an optically similar result in Swing, and it would be a development project of its own in AWT. In HIP, everything is at your fingertips. As the mask can also contain gradients, any user interface design idea can be realized.

The HIP classes are accompanied by a rich set of methods and properties, making GUI development a lot easier. Next to the makeDraggable method are float value and percent setters for the transparency. In Figure 4, I've set the transparency value of an OWLJPanel to 40%, causing it (and all its children) to be rendered with a 40% transparency, so the parent component shines through the panel. The code is simple:

```
panel.setTransparency
Percentage(40)
```

FIGURE 1 Free-form crosshair

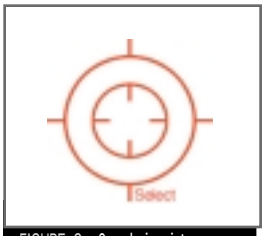


FIGURE 2 Crosshair picture

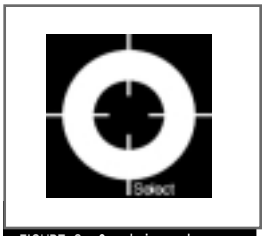


FIGURE 3 Crosshair mask

HIP J2 is worth using since transparencies and free-form shapes are a no-go in Swing; however, HIP PJ can bring these features even to PersonalJava. With HIP PJ you can develop advanced GUIs for communicators and handheld devices like a Compaq iPaq.

One thing that would make it even easier to start with HIP: additional ready-to-use code snippets. On the other hand, you can always go to OWL's online discussion boards for tips, tricks, and (free) support.

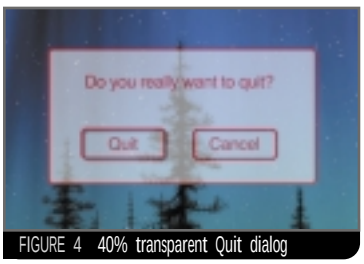


FIGURE 4 40% transparent Quit dialog

## Summary

Our World Live's HIP is targeted at those who develop their user interfaces in Java. Its unique features and full-fledged API sets are impressive. Transparencies and free-form shapes are the way to go in user interfaces, but were difficult to achieve in Java – until now. The API set is large and well thought-out. Everything has been modeled clearly and in accordance with the standard Java programming guidelines, so getting started is easy. The size of the frameworks is small, but be aware that your user interface graphics need space too. You can include your skin in your application or applet (which can get pretty big) or load it dynamically with a URL.

In addition to offering a whole new world of possibilities for GUI development, HIP can also reduce the time and cost for traditional user interface developments due to its rich set of methods and properties.

**JDJ Product Snapshot**

- **Target Audience:** Java programmers, project leaders
- **Level:** Beginner to advanced
- **Pros:** Unique features, easy-to-use and powerful APIs
- **Cons:** Nothing significant

# JBuilder 7.0 Enterprise Edition

by Borland

REVIEWED BY AJIT SAGAR ajit@sys-con.com

info:

**Borland**
100 Enterprise Way
Scotts Valley, CA 95066-3249
Phone: 831 431-1000
**E-mail:**
customer_service@borland.com
**Web:** www.Borland.com

**Specifications**
**Platforms:** Windows 2000/NT/XP, Linux and Solaris, Mac OS X

**Test Environment**
**Computer:** Sony Vaio Laptop
**Processors:** 1GHz Intel
**Memory:** 512MB RAM
**Platform:** Windows 2000, SP 2

There's no doubt about it. Borland makes great products for developers. They're definitely expensive and usually complex – but very powerful. I've been using JBuilder 6 for several months, and when I had the opportunity to review the latest version, I jumped at the chance. I won't take up much of your time by comparing this version with earlier ones, although I will definitely highlight the new features that I found most useful. This review focuses on the Enterprise Edition of JBuilder, not the Standard or Professional.

To do serious Java development, especially in a project team of more than two developers, you need a Java IDE. And you need one that provides features that will free your development life from mundane, tedious, and error-prone tasks. The following are some of the criteria I used to evaluate JBuilder 7.0:

- Ease of installation and setup
- Flexible and intuitive project setup
- Support for multiple development-projects
- Powerful code-editing features
- Ability to visually inspect class design
- Facilities for deploying a distributed application, including deploying to leading application servers
- Support for refactoring techniques
- Support for multiple JVMs
- Support for generating test classes for all types of Java components (JSP, servlets, Java classes, EJBs)
- Support for JSP editing and viewing
- Powerful debugger
- Flexible project handling
- Support for version control systems

I found JBuilder 7.0 met most of my criteria. What follows is a summary of the results of my evaluation.

## Product Description

JBuilder is one of the leading IDEs for Java development, and the Enterprise version is used to build distributed J2EE-based applications. In addition, version 7 includes the option to download a Web services pack for developing Web services from your Java components.

## Installation and Setup

Installation took 10 minutes. JBuilder's installation is based on Zero G Software and was very smooth. Since I wasn't running short of disk space, as is my standard policy, I selected the full install. However, starting the IDE cost me another 15 minutes since I had to go on the Internet to register my license. This was a minor annoyance, as I wasn't connected to the Internet when I installed the product.

Setting up the project was fairly easy (see Figure 1). I also found that there's a lot of flexibility in configuring the source directories, including libraries and setting up test directories. One of the most useful features is the ability to work on two different projects simultaneously. JBuilder lets you open multiple IDE browsers, one for each project (see Figure 2).

## Menus

JBuilder has a plethora of useful tabbed menus, and describing each one in detail is beyond the scope of this review. However, I will highlight some of the features that I found useful. Figure 2 shows the tabs with an expanded Tools menu in one project and an expanded Wizards menu in another. The navigation through the menus is very intuitive. Besides the tabs, the coding area and the Object Explorer bring up the appropriate menus when you right-click. The window on the bottom left gives a granular view of the class being coded and also continuously refreshes with errors as you type. The following are some of the useful and distinguishing capabilities of JBuilder 7:

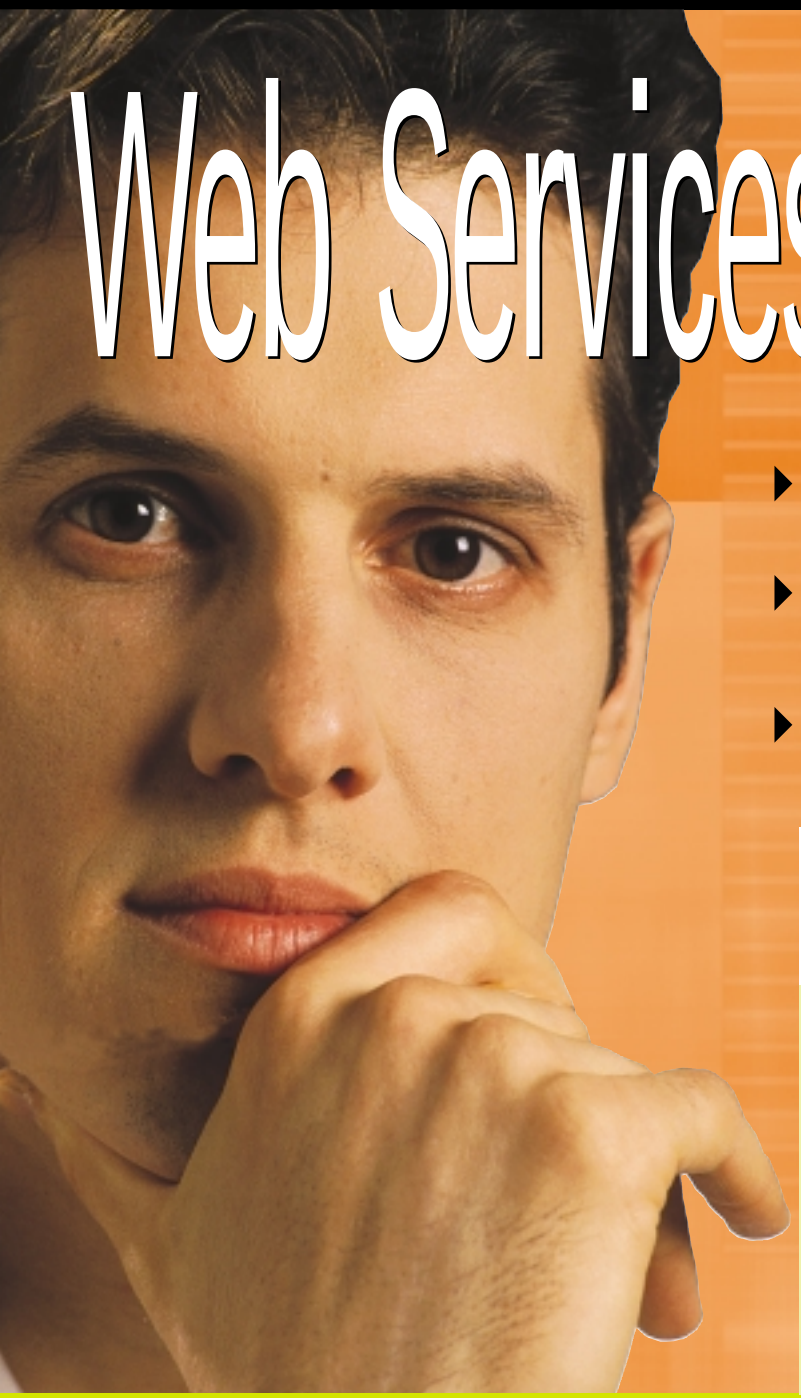
- It has wizards and tools for configuring deployment to application servers including BEA WebLogic, iPlanet, IBM WebSphere, and Tomcat. My applications are built on WebLogic, and it was very easy to set them up for EJB compilation and deployment.
- The Team menu provides direct access to source control systems. I set this up to work with VSS, our source code repository.
- JBuilder 7 is well integrated into supporting flexibility in build processes. I set up the project for external builds using Ant.
- It has an Object Gallery from the File menu (select New) for creating servlets, JSPs, EJBs, XML, Java classes, test classes, etc. I used the option for pulling in EJB definitions from an existing deployment descriptor and it worked like a charm.
- JBuilder 7 includes EJB development wizards with support for EJB 2.0 BMP.
- The File menu includes an option, Compare Files, to compare two source files and to merge changes between them.

Figure 3 shows the Object Gallery for Enterprise components, the app server configuration window, and the EJB Test Client generator wizard.

## The Code Editor

The code editor is the meat of the IDE. Besides code completion, parenthesis matching, smart indentation, and other neat editing gimmicks (which are very useful), I found JBuilder's editing capabilities far superior to most other IDEs in the market today. A right-click in the main editing window allows you to set up and configure the editor to your heart's

# Web Services Edge 2003



### Focus on Web Services

Companies that get an early jump on Web Services will be winners in today's challenging landscape. Get ready to take your early pilot projects to the next level!

### Focus on Java, XML, and .NET

Hear from the leading minds in Java how this essential technology offers robust solutions to *i*-technology. Make the right choices. Explore the evolving world of standards, interoperability, application integration, security, and more as you build a collaborative enterprise.

### Focus on the Knowledge You Need

Immerse yourself in information-packed conference sessions. Meet today's *i*-technology leaders, and gather resources in an action-packed Expo!

web services conference EDGE &expo

## The Largest

Web Services,
Java, XML, and .NET
Conference and Expo!

## Fall 2002

New York City


## THE NEXT FRONTIER:

BEYOND THE FIREWALL


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

JAVA DEVELOPER'S JOURNAL · WebServices JOURNAL · XML JOURNAL · BEA WebLogic DEVELOPER'S JOURNAL · ColdFusion Developer's Journal · PowerBuilder Journal · wireless BUSINESS&TECHNOLOGY · .NET DEVELOPER'S JOURNAL · WebSphere DEVELOPER'S JOURNAL · LINUX BUSINESS WEEK · CE Advisor

Call for Papers Opens
October 15, 2002

201 802-3069

content. As shown in Figure 2, I set up the colors to my favorite theme. Shortcuts and meta keys can be completely reconfigured in a manner that Emacs users would be proud of. Other useful features I found included the ability to set up bookmarks, code insight options, and coding style.

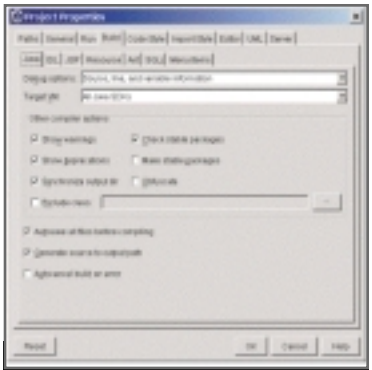


FIGURE 1 Setting up a project in Project Properties

At the bottom of the main editing window is a set of tabs that made development so much easier. In a class design window, the tabs Source, Design, Bean, UML, Doc, and History are available (see Figure 2). This enables you to switch from writing code for designer wizards to writing code for Javadocs, UML-style class diagrams, or your source control system – while in the same window. The UML-based feature was most useful in documenting code and understanding existing code. The diagrams are linked via clickable references that allow you to navigate between classes. The shortcomings of the UML features are that there's no way to get the overall class diagram (you can only view one class at a time) and the diagrams can only be saved in .png format.

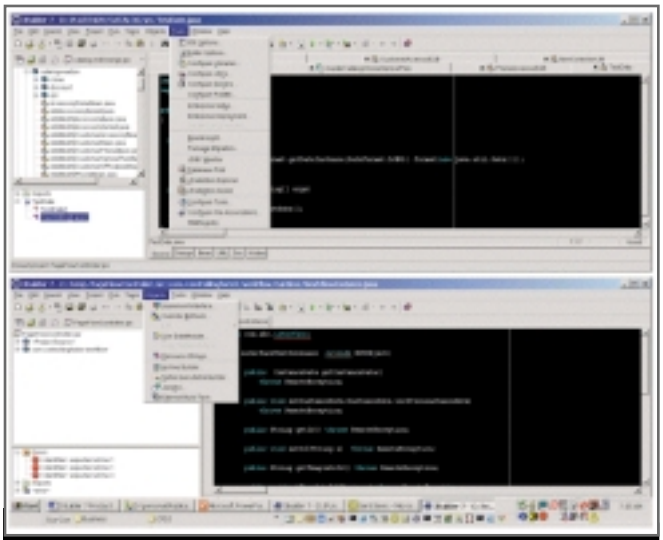


FIGURE 2 Working with multiple projects in multiple browsers

## Refactoring

JBuilder 7 provides good support for refactoring, including finding references and definitions; optimizing imports; moving, renaming, and repackaging classes; surrounding existing code with try-catch blocks (very useful!); and extracting methods from existing code. These options are available through the right-click menu. The relevant ones can also be accessed via right-clicks in the Object Explorer window. I've used each of these features at some point in the development cycle. The refactoring support makes redesign and coding much easier.

I could go on and on about the neat features that JBuilder provides. I've probably used about 20% of the functionality in its arsenal (remember the 80-20 rule). However, such details are beyond the scope of this review. In a nutshell, JBuilder is a great IDE for Java enterprise development that saved me a lot of development and debugging time in my projects while allowing me to manage the entire life cycle of my application.

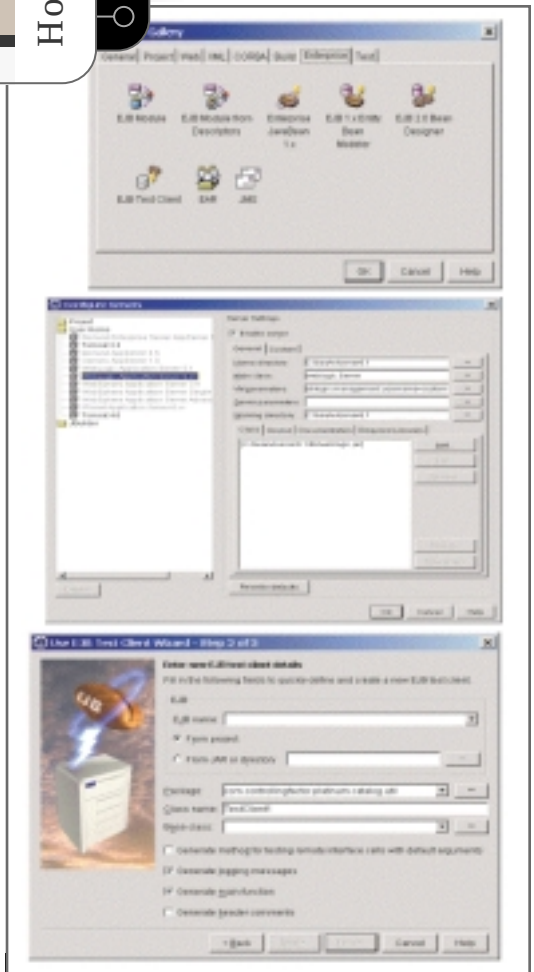


FIGURE 3 JBuilder 7 features

### Borland JBuilder 7.0 Snapshot

**Target Audience:** Enterprise Java developers

**Level:** Medium to expert

**Pros:**

- Excellent editing features
- Robust support for design and deployment of J2EE components, including EJBs, JavaBeans, Java servlets, and JSPs; visual EJB designer
- Powerful J2EE component development wizards
- Deployment support for popular application servers
- Option to automatically generate Javadocs within the IDE
- Excellent project management
- Good support for refactoring
- Excellent user interface, including intuitive and easy-to-use menus; visual designer for switching between source code and visual representation
- Robust support for databases through DataExpress
- Support for Ant builds and CVS, ClearCase, and VSS for source control
- Support for XML parsing
- Support for JDK 1.4
- Support for Web services via a downloadable Web services kit

**Cons:**

- Multiple layers of panes require large real estate on the screen. It's very easy to open up too many windows, which leads to confusion
- JBuilder is expensive as compared to other functional IDEs with similar features

TO REGISTER: www.sys-con.com or Call 201 802-3069

$195
FOR SYS-CON SUBSCRIBERS
BEST EDUCATIONAL VALUE OF THE YEAR!

web services world tour EDGE 2002

# Take Your Career to the Next Level!

EACH CITY WILL BE SPONSORED BY A LEADING WEB SERVICES COMPANY

## Learn How to Create, Test and Deploy Enterprise-Class Web Services Applications

### SHARPEN YOUR PROFESSIONAL SKILLS.

### KEEP UP WITH THE TECHNOLOGY EVOLUTION!

"Presented an excellent overview of Web services. Great inside knowledge of both the new and old protocols. Great insight into the code piece."
*– Rodrigo Frontecilla*

"Very articulate on the Web services SOAP topic and well-prepared for many questions. I've learned a lot from this seminar and I appreciate this seminar for my job. Thank you!"
*– Kenneth Unpingco, Southern Wine & Spirits of America*

"I liked the overview of Web services and the use of specific tools to display ways to distribute Web services. Good for getting up to speed on the concepts."
*– B. Ashton, Stopjetlag.com*

**Echoed over and over by Web Services Edge World Tour Attendees:**

"Good balance of theory and demonstration."

"Excellent scope and depth for my background at this time. Use of examples was good."

"It was tailored toward my needs as a novice to SOAP Web services – and they explained everything."

### WHO SHOULD ATTEND:

- **Architects**
- **Developers**
- **Programmers**
- **IS/IT Managers**
- **C-Level Executives**
- ***i*-Technology Professionals**

**TAUGHT BY THE INNOVATORS AND THOUGHT LEADERS IN WEB SERVICES**

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASIC TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI AND XML, PLUS MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS AND REMOTE REFERENCES.

**SPONSOR A CITY!**
Position your company as a leader in Web services
**Call 201 802.3066 to discuss how**

### IF YOU MISSED THESE...

**BOSTON, MA** (Boston Marriott Newton) SOLD OUT!
**WASHINGTON, DC** (Tysons Corner Marriott) SOLD OUT!
**NEW YORK, NY** (Doubletree Guest Suites) SOLD OUT!
**SAN FRANCISCO, CA** (Marriott San Francisco) SOLD OUT! CLASSES ADDED

### BE SURE NOT TO MISS THESE...

### ...COMING TO A CITY NEAR YOU

2002

| | |
|---|---|
| SAN JOSE | OCTOBER 3 |
| LOS ANGELES | NOVEMBER 5 |
| NEW YORK | NOVEMBER 18 |
| SAN FRANCISCO | DECEMBER 3 |
| BOSTON | DECEMBER 12 |

2003

| | |
|---|---|
| CHARLOTTE | JANUARY 7 |
| MIAMI | JANUARY 14 |
| DALLAS | FEBRUARY 4 |
| BALTIMORE | FEBRUARY 20 |
| BOSTON | MARCH 11 |
| CHICAGO | APRIL 16 |
| ATLANTA | MAY 13 |
| MINNEAPOLIS | JUNE 10 |

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE LOWEST REGISTRATION FEE.

PRESENT YOUR COMPANY'S EXPERTS TO AN EAGER AUDIENCE READY TO LEARN FROM YOU! ACT TODAY!

**TOPICS HAVE INCLUDED:** Developing SOAP Web Services, Architecting J2EE Web Services

The San Francisco tutorial drew a record 601 registrations.

SYS-CON *i*-TECHNOLOGY EDUCATION

REGISTRATION FOR EACH CITY CLOSES THREE BUSINESS DAYS BEFORE EACH TUTORIAL DATE. DON'T DELAY. SEATING IS LIMITED.
**NON-SUBSCRIBERS:** REGISTER FOR $245 AND RECEIVE THREE FREE ONE-YEAR SUBSCRIPTIONS TO *WEB SERVICES JOURNAL*, *JAVA DEVELOPER'S JOURNAL*, AND *XML-JOURNAL*, PLUS YOUR CHOICE OF *BEA WEBLOGIC DEVELOPER'S JOURNAL* OR *WEBSPHERE DEVELOPER'S JOURNAL*, A $345 VALUE!

TO REGISTER:
www.sys-con.com
or Call 201 802-3069

### Evolution Hosting Offers WebLogic in Flagship Platform

*(Boulder, CO)* – Evolution Hosting's core offering, the Evolution Hosting Platform, now supports WebLogic Server. The platform automatically configures application, database, and Web servers in your data center, alleviating common production issues that plague enterprise Java environments, such as account setup, resource connectivity, security, and management of server properties.

www.evolutionhosting.com

### BEA and Red Hat to Deliver Java Performance for Linux

*(San Jose, CA)* – BEA Systems, Inc., and Red Hat, Inc., have announced an expanded partnership to bring BEA WebLogic JRockit and WebLogic Server to Red Hat Linux Advanced Server customers.

According to the agreement BEA will certify its Java Virtual Machine, WebLogic JRockit, and WebLogic Server to run on Red Hat Linux Advanced Server. In turn, Red Hat will make WebLogic JRockit available to its customers through the Red Hat Network, which allows organizations to manage and deploy updates to multiple systems simultaneously.

www.bea.com
www.redhat.com

### CocoBase Enterprise O/R Now Includes JMS Caching

*(San Francisco)* – THOUGHT Inc. has announced that the newly released version of CocoBase Enterprise O/R includes expanded options for managing data with Distributed Caching based on the Java Message Service (JMS) specification from Sun Microsystems.

According to THOUGHT Inc., CocoBase JMS Caching can increase performance up to and over 20,000% versus using raw JDBC code with no caching. The caching solutions for CocoBase come as source code and can be modified as needed by the developer.

www.thoughtinc.com

### Aplix, Symbian, and Tao Group Join EEMBC Java Group

*(El Dorado Hills, CA)* – Aplix, Symbian, and Tao Group are the newest members of the EEMBC (Embedded Microprocessor Benchmark Consortium) Java Subcommittee.

EEMBC's Java group is developing a new series of benchmarks focused on the J2ME platform that will allow objective and certifiable performance evaluations for Java Virtual Machines (JVMs), just-in-time Java compilers, Java accelerators and coprocessors, other Java execution mechanisms, and the interaction of all of the above with a host operating system in an embedded application.

www.eembc.org

### nGame and Sprint Launch J2ME Games Through PCS Vision

*(San Francisco / Overland Park, KS)* – nGame Inc., a publisher and content aggregator of games for Internet-enabled wireless phones in North America, and Sprint have launched seven J2ME games on the enhanced Sprint Nationwide PCS Network. The full-color games are a part of the new PCS Vision, a next-generation wireless service recently launched nationwide by Sprint.

Zip:Sprint will be the first carrier to offer nGame's color J2ME games that will include *SLAM*!, the grand prize winner of the J2ME application design contest at the 2001 PCS Developer's Conference. Other titles available at the launch include *Aerostorm* and *Push Mania* from Korean Developer MobileGame and nGame's four-in-a-row and casino game titles *Riverboat Blackjack*, *Cash Quest Roulette*, and *Treasure Chest Slots*.

www.ngame.com
www.sprint.com

### Oak Grove Systems Releases Reactor 5.0.7 Workflow Engine

*(Calabasas, CA)* – Software developer Oak Grove Systems Inc., provider of embedded workflow engines, has announced the release of Reactor 5.0.7, a J2EE-based, XML-driven, and Web services–enabled workflow engine offering universal compatibility with the latest releases of all major development platforms, application servers, and databases. It also features improved monitoring capabilities.

## NEXTEL, SUN, MOTOROLA LAUNCH NATIONWIDE 'UNIVERSITY WIRELESS DEVELOPER CONTEST'

*(Reston, VA / Plantation, FL)* – Nextel Communications Inc., Sun Microsystems, Inc., and Motorola, Inc., are offering more than $45,000 in scholarships or grants, wireless handsets, and prizes to students who develop the best J2ME applications specifically tailored to make university life easier.

The first-ever nationwide "University Wireless Developer Contest" runs from September through November 1, 2002. Contestants will be judged on the relevance of the application for the university market, sophistication of the application, and the business model.

For more information visit http://developer.nextel.com.

## SUN INTRODUCES LX50 ENTRY-LEVEL SERVER

*(San Francisco)* – Sun Microsystems has unveiled the Sun LX50, an x86-architecture entry-level server loaded with Sun and open-source software, the Sun Open Net Environment (Sun ONE) software stack, the customer's choice of Sun's new enterprise-ready Linux or the Solaris Operating Environment (OE), and a full suite of support services.

The enterprise-ready Sun LX50 server lowers cost of ownership and helps fill the security and stability gap that 32-bit Windows-based systems leave open. It includes fully integrated infrastructure software, improved manageability, and a single source for hardware and software support and professional services.

http://sun.com

# JAVA JUSTICE DELAYED IS JAVA JUSTICE DENIED

***JAVALAND CALLS FOR MS TO PAY THE PRICE***

*by JDJ News Desk*

**IT WOULD BE HARD TO IMAGINE** a more emotive subject among developers than the question of whether Microsoft Corporation did or did not conspire to "kill off" Java on the client side five years ago.

From France, Germany, Romania, India, the UK, and all over America, Java developers have been responding to JavaLobby founder Rick Ross in his call for retrospective justice to be served on MS, in the Sun versus Microsoft case that's unfolding in the courts.

### *'I Didn't Know It Was Dead'*

Some developers take issue right away with Rick Ross's cry for justice. "Truly pathetic," writes Steve Vago. "I'm a member of JavaLobby and an enthusiastic Java developer, but this article is a pathetic rant. MS killed Java? I didn't know it was dead. Sun needs to concentrate on putting out good products and stop running to the courts for redress." Frank McGrath agrees with Vago. "What utter nonsense," he writes. "Some foolish Java zealots discuss Java as the be-all, end-all, the alpha and the omega."

"Obviously, if another language still lives, life is unfair," he continues, ironically. "If all OSs are not relegated to being low-level file systems, someone (probably Microsoft!) is cheating."

"What does Rick Ross mean by Java anyway?" McGrath asks. "Is it a language? Then it has serious competition from C++, Smalltalk, C#, VB, and many other languages. In what way has Microsoft inhibited its use as a language? Because Microsoft tried to extend the core Java language? What is the harm in that? What foolishness. What other vendor in the history of software has tried as hard as Sun to prevent others from extending the core of a software language? Decreeing that such extensions are nonstandard would be fine, insisting that the extensions be clearly described and documented as nonstandard would be fine, but that should be the end of it. Comparing the effort to 'extend and embrace' a computer language or environment to *murder*? This is insanity."

### *'Worse Than Killing'*

William Volk is for Rick Ross rather than against. Indeed he thinks that an even worse crime than a five-year-old murder has been committed: namely, present and ongoing deception. "What's worse than killing Java is how some of the XP systems are lying about the presence of Java. To be specific," he continues, "we test for Java via this JavaScript statement (after first checking that JavaScript is there): if (navigator.javaEnabled()).

What happens on some XP machines is that they'll tell us they do have Java…when they *don't*…and then they'll prompt the user to see if they want to download the Java plugin (sometimes). A bad thing. This is worse than killing off Java." ✎

To read more letters or to respond to this article, go to www.sys-con.com/java.

Reactor 5.0.7 is available as source code on a nonexclusive, royalty-free basis for a one-time, up-front fee.
www.oakgrovesystems.com

### FatWire Software Launches UE Studio 2.0

*(Mineola, NY)* – FatWire Software has announced the release of UE Studio 2.0, the latest edition of its development tool to build and automate Web applications.

An add-on to FatWire's enterprise-level dCM software UpdateEngine, UE Studio 2.0 offers expanded functionality to maximize developer efficiency and ease the application backlog. New features include a JSP builder and Data Import Tool.
www.fatwire.com

### Graham Pimlott Joins SpiritSoft's Board of Directors as Chairman

*(Boston / London)* – SpiritSoft has announced that Graham Pimlott has joined the message integration company's board of directors as chairman. Pimlott will oversee the continued implementation of SpiritSoft's growth strategy and product roadmap, working closely with CEO Kevin Gibbs and the management team.
www.spiritsoft.com

### Teamstudio Improves Reliability of Java Applications

*(Beverly, MA)* – Teamstudio has released its newest tool, Teamstudio Analyzer for Java, a best-practices audit tool for testing, debugging, and documenting Java applications.

Analyzer for Java works directly from the developer's IDE, with initial support built in for Eclipse, Borland's JBuilder, and Ant. It's Teamstudio's first offering for Java developers and marks the inception of the company's Teamstudio for Java product line.
www.teamstudio.com/analyzerforjava

### Intel Licenses Precise Technology to Boost Performance of Java Applications

*(Westwood, MA)* – Intel Corporation has licensed Precise's J2EE collection technology from the Precise/Indepth for J2EE product and is incorporating it into an Intel software development product for enterprise, Java-based Web applications.

The VTune Enterprise Analyzer for Web Applications, Java Edition, helps J2EE developers and QA teams identify distributed Web application bottlenecks and validates the hardware and software requirements of the application. Performance data is primarily collected through Precise's J2EE collection technology, which introduces only a low level of intrusion and minimal interference with the target servers and the application. It provides extensive insights into how the Web infrastructure is utilized while serving the client's request.
www.precise.com

### Sonic Software Releases SonicXQ 1.5

*(Bedford, MA)* – Sonic Software has announced the availability of SonicXQ 1.5, with improved performance and scalability, and extended support for all major application platforms.

The SonicXQ enterprise service bus (ESB) is an enterprise-class integration infrastructure for distributed business environments and computing models.
www.sonicsoftware.com

## Excellent Open-Source Tools

I enjoyed reading Jason Bell's editorial "Desert Island Open-Source Disks" (Vol. 7, issue 8). Excellent choice of open source tools.

But you forgot to mention Eclipse! Eclipse is a fantastic IDE for Java development, and it's open source too.

**Renaud Waldura**
renaud@waldura.com

*I'm glad you enjoyed the article. I didn't forget Eclipse; it was just a struggle between that and JEdit. What finalized it for me was that I could download JEdit at home on a 56K modem and get working right away.*

*I have used Eclipse on and off, depending on what mood I'm in.*

*Jason Bell*
*jasonbell@sys-con.com*

## Identical Performance?

I tested the modulo versus decrementing counter technique, and it was 3–4 times faster to use a counter…but only in an empty loop ("Performance Tuning in Java" by Dov Kruger [Vol. 7, issue 8]). If the loop executed anything, such as Thread.sleep(1), the counter technique was almost identical to the modulo in performance. Any ideas why?


**Colin Angel**
colin_angel@yahoo.com

*Computers are pretty fast these days. Divisions are, as you saw, quite a bit slower than the several instructions to do the rest of the loop but, still, your computer is doing tens of millions (perhaps even hundreds of millions) of those divisions in a second. So if you try to Thread.sleep, even for a millisecond, that totally dominates the time. Incidentally, I have never tried to sleep for 1 millisecond, and since a PC's timer is accurate to 1/60 of a second, I would be surprised if it really was a millisecond. Not that it matters; the call no doubt takes a while.*


*Dov Kruger*
*dkruger@righttrak.com*

## Expanding Java Coding

Thanks for "Whole House Audio from the Palm of Your Hand" by Bill Ray (Vol. 7, issue 9). His project will help me learn the Audio API and give me an excuse to buy a PDA to expand my Java coding into the handheld realm.

I code Java as a hobby only, and I'm looking for a fun project to help me continue to learn JSPs and servlets. Is it feasible to set this project up with a Web page interface (using JSP/servlets) instead of using Swing? I have some experience with JSPs, but not with handhelds. I assume if I can connect the handheld to my local network with Bluetooth, I can access my Tomcat servlet page as an interface.

**Scott J Young**
via e-mail

*Glad you liked it, I enjoyed working on it and just got my doorbell integrated too (see Part 3 [Vol. 7, issue 10]).*

*First, there's no Swing on handhelds; we use the AWT. I've never liked Swing personally, too big and slow, which is one of the reasons I tend to work in J2ME now. Second, Bluetooth isn't really about TCP/IP networking. While it's possible to connect to a TCP/IP network over Bluetooth, it's not an elegant way to work.*

*You'll need to bridge between Bluetooth and your LAN, using either a software bridge (one is supplied with the TDK USB Bluetooth adapter) or a Bluetooth LAN Access Point, such as the Inventel I use.*

*Good luck, and let me know how it goes.*

*Bill Ray*
*bill@network23.co.uk*

## The Future Is Here

I enjoyed "The Computer of Tomorrow" by Jason Briggs (Vol. 7, issue 8), but a lot of your "wishful thinking" is already here – if you put in the effort. Yes, it would be nice if some vendor came out with nicely packaged and ready-to-use products, but all the components are available. My house is a good example. I'm able to "plug any device into a port in any room and have it all work instantly without any messy configuration or setup."

The reason you don't see these things today is not that it isn't possible; it's simply that the companies that make the products we all use are out to make a quick buck. Our convenience is not their concern, unless they can profit significantly and easily from it.

**Bruno Melloni**
bruno.melloni@nokia.com

*I agree, there's nothing especially new or even difficult about any of the ideas I mentioned – nothing that a bit of brainpower, hard work, and (above all) money couldn't sort out. My main focus was really from the point-of-view of your normal, nontechnical Joe Public; I imagine, if he or she tried to duplicate your example without some serious technical help would wind up with less on top than your average Mexican Hairless.*

*Jason R. Briggs*
*jasonbriggs@sys-con.com*

**sys-con**

www.sys-con.com

# Java Interviews—What Might They Ask Me?

## Java questions . . . by Java developers . . . for Java developers

A job interview can be scary at the best of times. In the current economy, the process is even more intimidating. What sort of questions would you expect? What level of discussion are you likely to enter into in a Java interview? You – the readers – responded with a wonderful array of questions.

***Warm Up Questions***

How many interfaces can an interface extend?

*—Hugh McBride*

Is Java fully object-oriented? If yes, why?

*—Yogaraj Jayaprakasam*

Explain garbage collection in Java and the impact upon memory usage and performance in two of the common algorithms for determining when the space allocated to objects can be returned to free storage.

*—Harry Kaplan*

What is the difference between wait and sleep and why is it important?

*—Martin Smith*

Can you store null in a Hashtable?

*—Fatih Karakurt*

How do you allocate memory in Java?

*—Ratan*

What are the four modes of access protection in Java, and how do they behave? What's special about "protected" access?

*—Bruce Wilson*

***Java as a Language***

Given the opportunity to dictate changes for the next Java release, what would you fix?

*—Brian Gentry*

How do you deprecate a method?

*—John Davies*

What is the difference between overloading a method and overriding a method?

*—Vikrant*

What is reflection? When would you use it?

*—Jill Qian*

How do inner classes impact OO design? Do they help create better OO designs?

*—Mohan Radhakrishnan*

***Design-Type Issues***

Assuming you know the philosophy of the Java object model, the syntax of the Java language, how to use the Java platform, and the semantics of the specific APIs that are pertinent to the job you are seeking – how do you decide what to do when confronted with a need for new functionality?

*—Lee Hall*

An interface doesn't contain any implementation. Why would you want to use it?

*—Dushyanth Inguva*

What's an immutable object? What are some examples? Why is immutability important in Java? How would you create an immutable class?

*—Bruce Wilson*

If you needed to hold a collection of objects within a class, how would you do it?

*—David Kavanagh*

How can you leverage the salient features of Java to create an extensible and pluggable component framework?

*—David Forden*

***Political Questions***

Where do you see Java in relation to other languages and development environments? What are its strengths?

*—Scott Frazor*

What three things would you most like to change about the Java language? Not API stuff like "make AWT/Swing faster," but rather language-level specifics such as the assertions that were recently added or the semantics of public/private/package visibility.

*—Eric Rizzo*

Assuming you'll only deploy on Windows platforms, give three reasons why you would choose Java instead of C#.

*—Jim Long*

***Summary***

These questions covered all facets of Java, and we presented a few here so you would know what to expect. If you recently had an interview, we'd love to know the sort of questions you were asked.

• • •

Access the full list at www.n-ary.com/java/jdj/askjdj_question.cfm?JDJ_ID=2.

# JAVA DEVELOPER'S JOURNAL ADVERTISER INDEX

| ADVERTISER | URL | PHONE | PAGE |
|---|---|---|---|
| AccelTree | www.acceltree.com | | 34 |
| Altova | www.altova.com | | 27 |
| AltoWeb | www.altoweb.com | | 23 |
| BEA | www.dev2dev.bea.com/useworkshop | | 9 |
| BEA dev2dev | www.bea.com/events/dev2devdays | 404-240-5506 | 67 |
| Borland Software Corp. | www.borland.com/new/javasolutions/92122.html | | 21 |
| Capella University | www.capella.edu/getcred | 888-CAPELLA ext. 6027 | 56 |
| Crystal Decisions | www.crystaldecisions.com/cr9/002/ | 888-333-6007 | 13 |
| Engenuity Technologies | www.jloox.com | 800-684-5669 | 37 |
| ESRI | www.esri.com/arcims | 888-289-5084 | 33 |
| Fiorano Software | www.fiorano.com | 800-663-3621 | 57 |
| HiT Software | www.hitsw.com | 408-345-4001 | 53 |
| IBM | ibm.com/websphere/ebaydev | | 31 |
| IBM | ibm.com/db2/win | | 43 |
| InetSoft Technology Corp. | www.inetsoft.com/jdj | 888-216-2353 | 55 |
| Infragistics, Inc. | www.infragistics.com | 800-231-8588 | 14-15 |
| iSavvix | http://community.isavvix.com | | 63 |
| Java Developer's Journal | www.sys-con.com/java | | 76, 77 |
| JBoss Group | www.jboss.org | | 36 |
| JDJ Store | www.jdjstore.com | | 79 |
| Jinfonet | www.jinfonet.com/jdj10.htm | 301-838-5560 | 25 |
| Macromedia | www.macromedia.com/go/jrun4jdj | | 45 |
| Motorola | www.motorola.com/developers/wireless | | 4 |
| /n software inc. | www.nsoftware.com | | 39 |
| n-ary | www.javaSOS.com | | 69 |
| Northwoods Software | www.nwoods.com/go/ | 800-434-9820 | 38 |
| ObjectFocus Inc. | www.objectfocus.com | 408-973-8200 | 64 |
| Oracle Corp. | www.oracle.com/ad | 800-633-1072 | 19 |
| Our World Live | www.ourworldlive.com | | Cover III |
| Parasoft Corporation | www.parasoft.com/jdj10 | 888-305-0041 | 35 |
| Precise Software | www.precise.com/jdj | 800-310-4777 | 51 |
| Primavera | www.primavera.com | | 80 |
| QUALCOMM Incorporated | www.qualcomm.com/brew | | 49 |
| Rational Software | www.rational.com/offer/javacd2 | | 11 |
| Sitraka | www.sitraka.com/jclass/jdj | 800-663-4723 | 17 |
| Sitraka | www.sitraka.com/jprobe/jdj | 800-663-4723 | 47 |
| Sitraka | www.sitraka.com/performasure/jdj | 800-663-4723 | Cover IV |
| Sonic Software | www.sonicsoftware.com/jdj | 800-989-3773 | Cover II |
| Spiritsoft | www.spiritsoft.com/climber | | 6 |
| WebAppCabaret | www.webappcabaret.com/jdj.jsp | | 54 |
| Web Services Edge 2003 | www.sys-con.com | 201-802-3069 | 71 |
| Web Services Edge World Tour | www.sys-con.com | 201-802-3069 | 73 |
| Zero G | www.zerog.com | 415-512-7771 | 3, 61 |



# What's in the next issue of *JDJ*?

## MANAGING JAVA SOURCE CODE DEPENDENCIES FOR SCM

There are many facets to consider in the implementation of even the most basic software configuration management (SCM). For Java, with its import mechanism, these simple goals often become unmanageable when the source code tree grows beyond a certain point of complexity. This article discusses the underlying relationships that make even basic Java SCM problematic and how to manage them.

### PLUG IN YOUR COMMAND PROCESSOR NOW AND START SAVING MONEY!

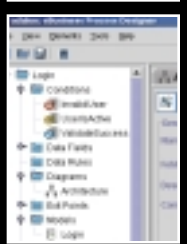
The Command Processor tool takes a Java object and creates a command-line interface to its public methods. These public methods are essentially your Application Programming Interface (API). During the course of this article we'll get a good look at the java.lang.reflect package, as well as kick the tires on the Regular Expression package included in the 1.4 JDK.

### UNLIMITED ENCRYPTION ON LIMITED DEVICES

A discussion about encryption in CLDC, focusing on one particular implementation

### MIDP 2.0

A look at the new features provided by the next iteration of the Mobile Information Device Profile



### *PRODUCT REVIEW* ADALON 2.2 BY SYNTHIS CORPORATION

Is there a development team out there who has never heard the following phrases when delivering a project? "That's not what we wanted…" or "We thought it would also do this…" Adalon 2.2 is a new type of tool specifically created to enable developers to accurately define and design complex Web-based software solutions from a business/functional perspective.

### *JDJ* READERS' CHOICE AWARDS

Find out the results of the *JDJ* Readers' Choice Awards. These awards, the "Oscars of the Software Industry," are given to superior products nominated and selected by a large number of professional Java developers and readers.

WRITTEN BY
BLAIR WYMAN

# Funk

**I'm sorry, dear reader, but I'm afraid I'm in a bit of a "blue funk" this month. There's an awful lot going on, it seems, and I don't understand most of it. I'm confused, angry, irritable, way behind in my work, and quite probably suffering from the after-effects of an alien abduction or some such.... What a drag.**

Take this column, for instance. For the life of me, I simply can't imagine that my few words here are going to make even a tiny bit of difference to anyone, anywhere, ever.

At best, I am probably just contributing to the world's ever-expanding quota of irrelevant and meaningless noise, doing my bit to lower society's cumulative signal-to-noise ratio even further below the audible threshold.

At worst, I imagine that my **Cubist Threads** are being used to torture captured enemy translators and aspiring magazine editors...

*Cut to a dank concrete dungeon, smelling of stagnant correction fluid and illuminated only by a single 100-watt bulb swinging from the acoustic-tiled ceiling. A figure is seated at an old formica desk, the only furniture in the room, as another figure raises a sheaf of papers high overhead as if to strike...*

"Edit *this*!" shouts the interrogator, spittle spewing from his sputtering spout, as he slams the first draft of my December, 2001, foosball column down on the gruesomely ink-stained table."...and be *quick* about it!"

"No! Oh, no! Please! I can't take another mixed metaphor or dangling participle from this, this *lunatic*! AIY-EEEE!"The hapless editor vainly struggles against the tightly wound Web of mailing labels strapping her to the stiff wooden chair.

"Only a blithering, certified, four-star *goofball* could come up with such a load of worthless hogwash!" she wails. Her eyes dart madly around the room, vainly searching for solace or escape; finding none, they flutter closed in a desperate swoon.

A dark figure, wearing a kilt, steps quietly from the shadows. Tightly clutching a handful of unsharpened #2 pencils, he motions the interrogator to move aside. "I'll take over," he says, "this situation calls for extreme measures."

The dark figure grasps one of his #2 pencils tightly, raises it ominously overhead, and viciously stabs it directly down into an old electric pencil sharpener, eraser end first.

The grinding of mangled metal snaps the swooning editor back to painful consciousness, but something is strangely different about her. An intense light of reason and clarity shines from her eyes. It is as though she swooned as one person, and returned as another – calmed by some Muse of Reason during her unconscious respite.

"All I have to do is edit this, right?" she asks. The dark figure nods, and hands over this month's Cubist Thread. The editor scans the text, smiles once at the ceiling, and quietly goes utterly and completely mad.

Wow. Where did *that* come from? My funk is apparently even deeper than I thought.

I guess I picked a bad week to quit smoking.

AUTHOR BIO
*Blair Wyman is a software engineer working for IBM in Rochester, Minnesota, home of the IBM iSeries.*

blair@blairwyman.com