

# JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

May 2002 Volume: 7 Issue: 5

JAVADEVELOPERSJOURNAL.COM

THE LARGEST WEB SERVICES CONFERENCE & EXPO IN THE WORLD!

web services **JDJEDGE** **XMLEDDGE** conference

SAVE \$200

Conference: June 24-27 Expo: June 25-27

The Future of Enterprise Computing

JACOB JAVITS

FULL CONFERENCE PROGRAM  
▶ INSIDE PAGE 99

From the Editor  
Alan Williamson pg. 7

J2EE Editorial  
Ajit Sagar pg. 9

J2SE Editorial  
Keith Brown pg. 36

J2ME Editorial  
Jason R. Briggs pg. 62

Industry Commentary  
Kuldip Singh Pabla pg. 64

JDJ Labs  
Enterprise JavaBeans pg. 90  
Java-Miner pg. 92










RETAILERS PLEASE DISPLAY  
UNTIL JULY 31, 2002

\$5.99US \$6.99CAN



SYS-CON MEDIA



	<b>Database Systems: Optimizing Database Performance in J2EE Apps</b> <i>The best of both worlds</i>	<b>Didier Cabannes</b>	<b>12</b>
	<b>Feature: The Critical Role of Application Architecture</b> <i>A fundamental issue</i>	<b>Walter Hurst</b>	<b>22</b>
	<b>Show Review: First Impressions JavaOne 2002: How did it measure up to past shows?</b>	<b>Ajit Sagar</b>	<b>30</b>
	<b>AI: Programming Neural Networks in Java</b> <i>An efficient way to perform certain operations</i>	<b>Jeff Heaton</b>	<b>38</b>
	<b>JDBC 3.0: Something for Everyone</b> <i>The new features and why they're important</i>	<b>John Goodson</b>	<b>56</b>
	<b>Jini &amp; J2ME: Jini Surrogate As a Platform for J2ME Games</b> <i>Using HTTP as a protocol</i>	<b>William Swaney</b>	<b>66</b>
	<b>Feature: OSGi: The Last Mile of Software Deployment</b> <i>Solve last-minute problems</i>	<b>Peter Kriens</b>	<b>74</b>
	<b>Pro Mobile: Keep Mobile Data and Applications in Sync with Java</b> <i>An integral part of your mobile strategy</i>	<b>Jeff Capone</b>	<b>84</b>
	<b>App Management: Manifest Destiny</b> <i>Simplify the packaging and releasing of Java applications</i>	<b>Norman Richards</b>	<b>94</b>

# Sonic Software

[www.sonicsoftware.com](http://www.sonicsoftware.com)

# Zero G

[www.zerog.com](http://www.zerog.com)

# Apple Computer, Inc.

[www.apple.com/macosx](http://www.apple.com/macosx)

# Apple Computer, Inc.

[www.apple.com/macosx](http://www.apple.com/macosx)

# BEA

[www.bea.com/download](http://www.bea.com/download)

## INTERNATIONAL ADVISORY BOARD

- CALVIN ALSTIN (Lead Software Engineer, J2SE Linux Project, Sun Microsystems)
- JAMES DUNCAN DAVIDSON (JavaServlet API/JMP 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: KEITH BROWN
- 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 MATUSOV, GAIL SCHULTZ, JEAN CASSIDY
- ONLINE EDITOR: LIN GOETZ
- TECHNICAL EDITOR: BAHADIR KARUV, PH.D.

## WRITERS IN THIS ISSUE

- BILL BALOGLU, JASON BELL, JASON R. BRIGGS, KEITH BROWN, DIDIER CABANNES, JEFF CAPONE, JOHN GOODSON, JEFF HEATON, WALTER HURST, PETER KRIENS, KULDIP SINGH PABLA, BILLY PALMIERI, NORMAN RICHARDS, AJIT SAGAR, HENDRIK SCHREIBER, BILL SWANEY, 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](mailto: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.

## © COPYRIGHT:

Copyright © 2002 by SYS-CON Publications, Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission. For promotional reprints, contact reprint coordinator Carrie Gebert, [carrieg@sys-con.com](mailto:carrieg@sys-con.com). SYS-CON Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.

Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. SYS-CON Publications, Inc., is independent of Sun Microsystems, Inc. All brand and product names used on these pages are trade names, service marks or trademarks of their respective companies.



ALAN WILLIAMSON EDITOR-IN-CHIEF

## My Kingdom for a Phone

The whole wireless space has been an interesting one to keep an eye on for the past couple of years, in particular how it relates to the Java space. Sure, we've heard wonderful tales about the vast millions of phones in Japan and how quickly Java is being adopted there, but for the rest of us here in the West, it's not quite as exciting. Regular readers know only too well my woes with my Nokia and the lack of Java support. The question is this: Is it a pipe dream or is it really coming?

That was the question I posed to Nokia at JavaOne. Whereby I was given the usual marketing pitch about the great array of developer's tools and shown the nice color chart of all the models that have been Java-enabled. Naturally, I ooh'd and ah'd in the right places. I then asked to what extent would the KVM be integrated. For example, would I be able to access the hundreds of telephone numbers already stored in my Nokia address book? The answer was no! Excuse me? Okay, try not to panic, I thought, not a major problem. I just need to import them to my Java application and they can reside there. I asked for confirmation that my Java application would be able to make phone calls from my now imported contact list. No was the answer again, "That would be a security risk and violate the openness of the specification."

At this point, I couldn't believe what I was hearing. Is this true? We have a phone...whose main purpose in life is to make calls. That's its calling, to use a very bad pun. We can load applications on it that can't actually take advantage of the core functionality of the phone? Major oversight, wouldn't you say? Tell me again why I want my next phone to be Java-enabled? Someone please e-mail me and tell me I'm wrong.

At JavaOne, I had a great meeting with Bruce Scott from PointBase; he kindly spent time discussing what they were doing and where they see the future of J2ME. For those

of you who don't know the name, Bruce was the main engineering man behind the database at Oracle, so when he talks about databases, you kinda wanna listen.

PointBase was doing some cool stuff with syncing technology, as they had given up hope regarding the always-on network, and their clients were looking for real-world solutions they could use today. It was a great JDBC database/driver that operated at the J2ME level. You would simply tag the table columns you wanted synced with the database at the back end, and when the network was available, the two would automatically sync with one another, recovering should the network drop. It doesn't seem that impressive on the face of it, until you realize they do all this in under 50KB! However, the reason I mention PointBase is that they were atypical of the shift in the wireless space I witnessed at JavaOne.

Developers were going to great lengths to tell me about their solutions regarding the state of the network, and the conditions their software would excel in, such as a temperamental network going up and down. This I found reassuring because I live in a rural area where reception for my mobile is not always 100%; I was beginning to worry that I may be left behind in this new always-on, always-connected, Web-serviced world!

On the whole, JavaOne was a good show with plenty of walking space between the booths (you definitely knew the bottom had fallen out of the dot-com world). I caught up with a lot of old faces, listened attentively to what they were up to, had a brush with J2EE Blueprints, and met a vast array of new faces. It was a great event, as usual.

We were there with SYS-CON Radio interviewing the latest and greatest. Keith Brown, our esteemed J2SE editor, conducted the majority of the interviews, which you can listen to online at [www.sys-con.com/java/](http://www.sys-con.com/java/).

Until next month ... ☎

## AUTHOR BIO

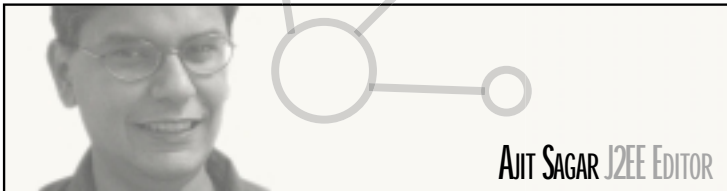
Alan Williamson is editor-in-chief of Java Developer's Journal. During the day he holds the post of chief technical officer at n-ary (consulting) Ltd, one of the first companies in the UK to specialize in Java at the server side. Rumor has it he welcomes all suggestions and comments.

[alan@sys-con.com](mailto:alan@sys-con.com)

# TogetherSoft Corporation

[www.togethersoft.com/1/jd.jsp](http://www.togethersoft.com/1/jd.jsp)





AJIT SAGAR J2EE EDITOR

# Are You Being Served?

I'll never buy a Casio watch again. Not just because they break down – that's just the luck of the draw – but because of their extremely poor service. I've spent months trying to get my \$200 watch back from their service center, but to no avail. It isn't the money that matters; it's the principle of the thing. For a couple of months it was nearly impossible to get anyone on the phone, and there was no advertised Web address for contacting the service center.

Customer service and relationship management is the cornerstone of a successful business. Today's software industry is a lot like other industries – it's service-driven and highly competitive. With the three-month Internet-year release cycle, vendors work hard to keep their service offerings at an acceptable level and to keep up with the rapid changes to the baseline products. They struggle with the same issues that all other industries do – selling products to customers and then retaining those customers. Vendor products that offer functionality for J2EE have to work on a framework that's evolving at a breakneck speed.

Since they deal with new functionality and APIs that are still stabilizing with every release of the platform, vendors have to make sure their marketing message is well coordinated with their ability to deliver the goods in light of rapidly evolving standards. Vendors face a paradoxical situation. To gain a larger market share, they need to make sure they're standards-compliant. To distinguish themselves from the competition, they need to offer proprietary value-adds to the basic functionality offered by Sun's specs for the J2EE platform.

Retaining customers is largely dependent on how well the products are supported. For the past few years, J2EE vendors

have been playing catch-up with Sun's J2EE platform APIs. The platform is now stable and there are ample examples of real-world deployments. At this stage in a platform's development cycle, support and service for J2EE-based products becomes of paramount importance. With the recent economic debacle, there's less room in organizations for "buy and try" and more of a requirement for well-supported products.

Before selecting a vendor who supplies J2EE wares, organizations need to make sure that adequate support will be available when they're struggling with critical and complex issues. Almost all the J2EE application server vendors have their partner programs. However, based on recent experience, I'd like to caution you before marrying your development to a specific product. It's very important to make sure support will be provided by skillful personnel in a timely manner.

One of the risks with the programs offered by vendors is that often the support person at the other end is not very qualified, and may be even less familiar with the product than you are. There's no option but to run the gauntlet before the support call is escalated to where you get the right level of technical support. Sometimes you end up educating the person at the other end before getting any valuable feedback. This could take several weeks and potentially jeopardize your deadlines.

To select the right vendors, it's important to look at their legacy. After all, J2EE is a fairly new platform – less than 10 years old. Therefore, most of the J2EE application server and other tool vendors are reincarnations of vendors who came from other product lines. The legacy defines their areas of expertise. Eventually, it determines the level of support they can offer to your application development. ●

ajit@sys-con.com

## AUTHOR BIO

Ajit Sagar is the J2EE editor of JDJ and the founding editor and editor-in-chief of XML-Journal. A lead architect with Metavonni, LC, based in Dallas, he's well versed in Java, Web, and XML technologies.

## Are You Being Served?

Customer service and relationship management is the cornerstone of a successful business. Before selecting a vendor who supplies J2EE wares, make sure that adequate support will be available when you're struggling with critical and complex issues.

by Ajit Sagar

9

## J2EE FAQ

The answers to your J2EE questions

10

## Optimizing Database Performance in J2EE Applications

With hybrid databases, J2EE developers can demand a database that meets the intrinsic requirements of scalability, high transaction volumes, high-volume data transfer, and the need for fast throughput as well as an object data model that more accurately represents business processes, now and in the future.

by Didier Cabannes

12

## The Critical Role of Application Architecture

Vendors are delivering tools and frameworks to ease the next level of application development for many project teams, allowing them to focus on the business requirements at hand and obviate the need to deal with what will soon become mundane development tasks. Nowhere will this be more apparent than in the area of application architecture.

By Walter Hurst

22

## JavaOne Show Review

If you didn't make it to JavaOne this year, find out what you missed – what the big announcements were, who the best keynote speakers were, and more.

by Ajit Sagar

30

## WHAT IS EJB 2.0?

**E**JB 2.0 is the latest release of the Enterprise JavaBean specification. The major releases of the EJB specification have been 1.0, 1.1, and 2.0. EJB 2.0 adds several crucial features to version 1.1, including message-driven beans, local interfaces, an enhanced container-managed persistence, and EJB-QL (Query Language).

## WHAT ARE THE MAIN INTERFACES IN EJB 2.0?

**T**he figure below illustrates the class hierarchy for EJB 2.0 and the interfaces that are included. The main interfaces for EJBs are packaged under `javax.ejb`; they're standard extensions to the core Java classes.

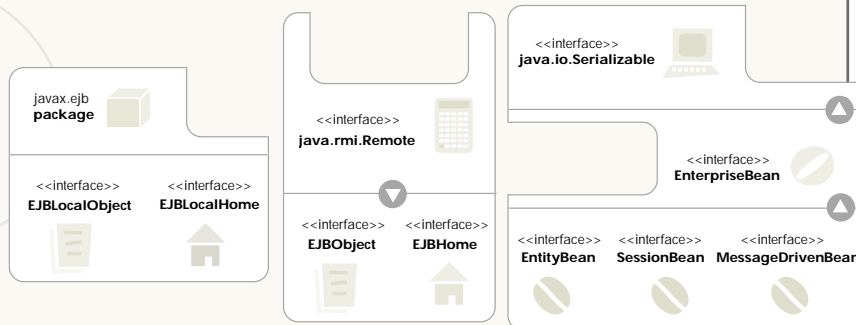
The `javax.ejb` package defines the interfaces you need to extend your application's components. By definition, Enterprise JavaBean is a specification for distributed architecture. In Java terms, this means that a Java class running in one JVM should be able to communicate with an EJB component in another via an RMI call. First, let's look at the `EJBObject` and `EJBHome` interfaces, which are shown in the middle pane of the figure. These interfaces extend the `java.rmi.Remote` interface. The `Remote` interface serves to identify interfaces whose methods may be invoked from a nonlocal virtual machine. Any object that's a remote object must implement this interface.

As you probably know, EJBs can only run inside a J2EE EJB container. The container abstracts many of the low-level services from the application developer (you). One set of services that it abstracts is the life-cycle management of the EJBs. The `EJBHome` interface is used to create this abstraction and can

(`EJBLocalObject` and `EJBLocalHome`) don't extend the `java.rmi.Remote` interface. You can't use them to access distributed objects. The local interfaces were added to the EJB specification in 2.0 so that component accesses to EJB components within the same JVM don't have to be done through RMI, which is very expensive since each call requires a distributed call.

The enterprise bean interfaces are illustrated on the left side of the figure. These interfaces were added to the EJB specification in 2.0. Local interfaces support the concept of collocated EJBs (local interfaces). The local interfaces (`EJBLocalObject` and `EJBLocalHome`) don't extend the `java.rmi.Remote` interface so you can't use them to access distributed objects. Other than that, the `EJBLocalObject` is the equivalent of the `EJBObject`, and the `EJBLocalHome` is the equivalent of `EJBHome`. Note that the same EJB component can implement both interfaces simultaneously, thus allowing remote access through RMI and local access through a local method call.

The right pane of the figure shows the interfaces used to create the actual implementation class. The



be accessed remotely to create or find the actual EJB component. The methods are invoked by the remote client, but are executed by the container.

The `EJBObject` interface is used to define the business methods for your EJB component. The interface defines methods to access the actual EJB class and to remove it when it's no longer needed. The `EJBObject` is a delegator interface that delegates the actual execution of the business objects to the enterprise bean.

The latest release of the EJB specification adds classes and interfaces to support the concept of collocated EJBs (local interfaces). As you can see in the left pane of the figure, the local interfaces

`EnterpriseBean` interface is a serializable interface for creating the EJB component. The three types of EJBs supported in 2.0 are `EntityBean`, `SessionBean`, and `MessageDrivenBean`. Each of these extends the `EntityBean` interface.

As an application developer, for each enterprise JavaBean in your application, you need to extend the appropriate home and remote interfaces (local/remote) and provide an implementation for the appropriate enterprise bean (`entity/session/message-drivenbean`).

Details on developing the interfaces and classes for your application's EJBs will be covered in subsequent FAQs. ☛

## 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

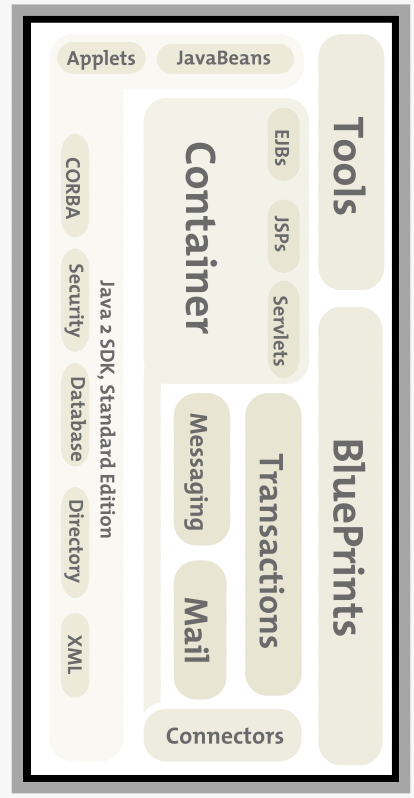
**Java Messaging Service** .....v. 1.0

Useful URLs:  
Java 2 Platform, Enterprise Edition  
[www.java.sun.com/j2ee/](http://www.java.sun.com/j2ee/)

J2EE Blueprints  
[www.java.sun.com/j2ee/blueprints](http://www.java.sun.com/j2ee/blueprints)

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

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



# Metrowerks Corp.

[www.wireless-studio.com](http://www.wireless-studio.com)

# Optimizing Database Performance in J2EE Applications

The best of both worlds



WRITTEN BY  
DIDIER CABANNES

**T**he Java 2 Platform, Enterprise Edition (J2EE), is the platform of choice for implementing scalable and reliable enterprise applications from reusable components. But Java developers building enterprise-class J2EE applications face a quandary.

The object paradigm has proven ideal for modeling a wide variety of real-world scenarios. However, finding a Java-compatible data repository optimized for such applications has become a stumbling block. While object database management systems (ODBMSs) provide the convenience of transparent persistence of Java objects, their client-centric architecture has not scaled well in enterprise environments. Relational database management systems (RDBMSs) do scale well, but map each object to a two-dimensional relational table. The increased overhead can reduce application performance to a crawl.

This article discusses the limits of using these two types of databases with Java and suggests a better alternative for J2EE – a hybrid database that combines the best features of both. Hybrid databases share with ODBMSs the ability to map data stored in back-end databases directly into an implementation-neutral Java representation. As with relational systems, hybrid databases can scale to meet the performance requirements of an enterprise-class J2EE application.

## ODBMSs: The Hidden Headache of Transparent Persistence

Over the years, finding a database that's both Java-compatible and scalable enough for enterprise-class J2EE applications has not been easy. Ideally, a Java-compatible database should store Java

objects whose classes have been declared "persistent-capable" and can be manipulated seamlessly by the Java language.

That has been the promise of ODBMSs, which made their appearance in the mid-1990s as a solution designed specifically for objects and thus better suited for object development. With ODBMSs, Java developers can define persistent Java classes in the same way transient Java classes are defined in the application.

An apparent advantage of pure object databases is the implementation of transparent persistence that automates the process of mapping persistent data objects into the data repository. With transparent persistence, you don't even have to alter your existing Java classes to describe the persistent data that's permanently stored in the database (see Listing 1). That means you don't have to decide ahead of time, usually during the design phase, which objects to include and exclude from the database.

Adding a new customer order into the database is as simple as creating a new object in Java. Persistent-capable objects are transient until attached to a persistent manager or to other persistent objects.

This convenience quickly becomes a nightmare, however, when developing scalable enterprise-class applications. In a typical application, objects are

highly interconnected, and it's very important to know precisely which objects have been stored with the database and which have not. Consider an e-commerce application in which products, customers, and orders are all linked together (see Figure 1). The object model naturally captures the interrelationships of real-world applications. With transparent persistence, you wind up loading an entire closure of objects even though you want to access only a single object (see Figure 2). While the programmer wants to load only one customer, the closure of instances reachable from this object recursively loads a large portion of the database. Loading unneeded data in the Java VM limits concurrency and scalability.

A simple customer query, for example, could also lock pending orders and products purchased, even though this data was not requested and will remain unchanged. Such "overloading" is not a noticeable problem within a standalone environment that manipulates a small amount of data. However, in an enterprise-class, multiuser, transaction-intensive application, large portions of data get locked and instantiated, limiting concurrency and scalability.

During the pilot phase of development, performance is usually acceptable since the system is not running under heavy computational loads. But with wider deployment and more users, transaction rates can slow unacceptably

# Sun Microsystems

[www.sun.com/forte](http://www.sun.com/forte)

# **Infragistics, Inc.**

[www.infragistics.com](http://www.infragistics.com)

# Infragistics, Inc.

[www.infragistics.com](http://www.infragistics.com)





# Sitraka

[www.sitraka.com/jclass/jdj](http://www.sitraka.com/jclass/jdj)

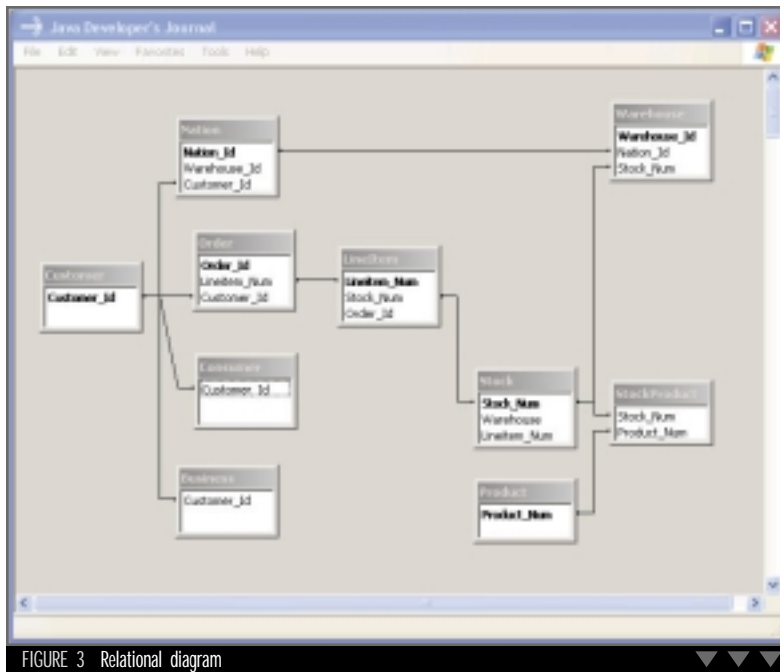


FIGURE 3 Relational diagram

number of OR mapping tools have been created. While these tools do make it easier to develop Java applications that use relational databases, they don't eliminate the underlying RDBMS problems of code complexity and poor performance.

Both database technologies have limitations for Java programming. A pure object database makes sense in a standalone environment in which concurrency and network traffic are not issues. Relational databases, while accommodating transaction-processing loads, merely simulate a true object environment.

### Hybrid Databases: The Best of Both Worlds

Hybrid databases represent the best of both worlds: the ability to map objects from Java directly to the database with the support of a standard query language (SQL-99) and the scalable, enterprise capabilities implemented in relational database products. Designed from the ground up as a database server for objects, hybrid databases directly map the object model of Java as well as other object programming languages. Because the database object model matches perfectly with Java, you can freely and easily define the database classes that describe real-world scenarios.

Unlike an RDBMS, a hybrid database preserves the original Java data model. For example, a single class and two subclasses represent customers, consumers, and business customers, respectively. No tables are mapped back into Java objects; no translation of any

kind is needed. Unlike an ODBMS, a hybrid database enforces a layered design of the persistent classes. The operations to manipulate objects are explicit, enabling you to keep tight control over the data that's locked and instantiated in the JVM, seamlessly improving the application's scalability.

Hybrid databases eliminate the mismatch between the Java and database environments, while still maintaining the scalability of server-side processing, such as relational systems. Within the J2EE environment, you manipulate Java objects representing a proxy to the object in the database by means of object-to-object mapping. The proxy objects are pure Java classes that map to those of the database schema (see Listing 3). With a hybrid database, the code stays compact and object-based (as in Listing 1), providing the same benefit as a first-generation ODBMS. Hybrid databases don't require any of the special compilation tricks or postprocessing byte code manipulations of ODBMSs – both of which make it hard to identify the root cause of performance degradation.

In a typical application, classes are highly interconnected, and the graph of instances can include large portions of the database. Therefore, controlling object-locking effectively, always a challenge in enterprise-class J2EE applications, is crucial to controlling the instantiation of Java objects in the JVM. To build scalable applications, data-intensive processing needs to take place where the data sits on the server, not on the client, further reducing lock-

ing contention as well as network traffic and taking advantage of the faster processing speeds of many server architectures.

Like RDBMSs, hybrid databases support the SQL-99 syntax. While SQL queries are relational in their syntax, they take advantage of the object paradigm by supporting inheritance, polymorphism, and true navigation. Furthermore, the query processing takes place on the server to enforce security and achieve performance. Consider a broad query of two classes of customers: business and consumer. The query is issued from the client, executed on the server, with selected objects from each class retrieved to the client.

This approach gives developers full access to Java objects through JDBC without having to learn a proprietary API (see Listing 4). In this listing, two customer subclasses, Consumer and Business, share properties from the parent Customer class while maintaining properties of their own. A query to locate "good customers" can combine criteria – bonus miles for home consumers, a high credit line for businesses – pulling the information simultaneously from both subclasses. Unlike an RDBMS, a hybrid database returns Java objects through JDBC and natively supports inheritance.

While developers still benefit from the power of expression and performance of SQL queries, these queries eliminate the object-relational mapping layer to reduce source code by 25–50% and improve application performance.

Unlike first-generation ODBMSs, hybrid databases can be accessed through JDBC and ODBC drivers, both of which support the SQL-99 language, thereby taking advantage of in-house SQL expertise. Support for ODBC and JDBC drivers also allows IT staff to use off-the-shelf database tools without having to master SQL.

### First Major Optimization: Keep It Simple

Building enterprise-class J2EE applications with a hybrid database is straightforward. Here are some considerations to make the process even smoother:

- Carefully define the object model of your persistent classes, reflecting the business model as closely as possible. That's common sense in an object environment, but is even more crucial in database applications because the way you define the model greatly impacts system performance.

# Oracle Corporation

[www.oracle.com/javacode](http://www.oracle.com/javacode)



J2ME



J2SE



J2EE



Home

Defining the right level of granularity for your objects has a big payoff in terms of transaction rate because only the specific queried data gets locked.

- Avoid cross-referencing persistent and transient objects as transient information can access persistent information, but not the other way around. Doing so makes the application much more complex to manage since the persistent objects loaded from the database may need to be linked to transient information that's not yet available. While a callback can also be used, it unnecessarily complicates program flow and can usually be avoided with more ordered layering of the application.
- Keep transactions as short as possible. Long transactions will unnecessarily lock data for long periods of time, making it unavailable to other business transactions.
- In some cases, data is cached by the middleware, reducing contention, but it requires "dirty reads" (reading data without locking) from the database. A way around this is to use a versioning facility, which allows a consistent view of the database any time, even while users are modifying the current version.

### Conclusion

Hybrid databases give developers a new and important option when selecting a database for their J2EE application. Until now, Java developers have really had just one viable option: an RDBMS. Despite the drawbacks of the relational model, only RDBMSs solved the performance requirements intrinsic to enterprise applications. As for ODBMSs, they haven't even begun to meet these requirements. Without that, an adaptable object model is irrelevant to large-scale J2EE development.

With hybrid databases, J2EE developers can demand both: a database that meets the intrinsic requirements of scalability, high transaction volumes, high-volume data transfer, and the need for fast throughput, together with an object data model that more accurately represents business processes, now and in the future.

As the number of J2EE applications grows, the limitations of RDBMSs and ODBMSs will become more and more apparent. Hybrid databases represent the missing ingredient for broader J2EE implementation, providing scalability without compromising Java's object environment. ☼

didier@fresher.com

#### Listing 1: Transparent persistence with an ODBMS

```
pmf.setConnectionURL(dbName);
pm = pmf.getPersistenceManager();
tx = pm.currentTransaction();
tx.begin();
Order odr = new Order("Parrot", 2, 99.95);
pm.makePersistent(odr); // make the Order persist
Customer cstr = new Customer("Jay", "D", "Ho");
odr.orderedBy = cstr ; // the new Customer now persists by the mean of
attachment
tx.commit();
pm.close();
```

#### Listing 2: The OR mapping layer adds 25-50% of ugly code

```
String select = "select * from Order o where o.odr_id = ?";
pstmt = conn.prepareStatement(select);
pstmt.setLong(1, 10608974);
resultSet = pstmt.executeQuery();
// Construct an instance of Order from its rows
if (resultSet.next()) {
    Order odr = new Order(resultSet.getLong("odr_id"),
        resultSet.getString("ship_address"),
        resultSet.getString("ship_carrier"));
}
resultSet.close();
pstmt.close();

String select = "select * from lineItem l where l.odr_id = ? order by
lt_id";
pstmt = conn.prepareStatement(select);
pstmt.setLong(1, 10608974);
resultSet = pstmt.executeQuery();
// Construct objects from rows
while (resultSet.next()) {
    LineItem litem = new LineItem(resultSet.getString("lt_id"),
        resultSet.getString("quantity"),
        resultSet.getBigDecimal("unit_price"),
        resultSet.getString("delivery_mode"),
        resultSet.getString("address"));
    odr.lineItems.add(litem);
}
resultSet.close();
pstmt.close();
```

#### Listing 3: A hybrid database provides compactness and efficiency

```
Database db("dbName");
db.open();
db.startTransaction();
Customer cstr = new Customer(db, "Dee", "O", "Haye");
Order odr = new Order(db, "Parrot", 1, 99.95);
odr.setOrderedBy(cstr) ; // the order and the customer are linked together
by a bi-directional relationship
db.commit();
db.close();
```

#### Listing 4: With a hybrid database, SQL queries return objects, not tables

```
String query = "select REF(c) from Customer c where count(orders) > 20" +
    "AND ((class Consumer).bonusMiles > 50000 OR" +
    "(class Business).creditLine > 10000) ORDER BY lastName";
stmt = conn.createStatement();
resultSet = stmt.executeQuery(select);
while (resultSet.next()) {
    Customer cstr = (Customer)resultSet.getObject(1); // instances of
Consumer or Business are actually returned
}
resultSet.close();
stmt.close();
```

Download the Code!  
www.javaDevelopersJournal.com

**AUTHOR BIO**  
Didier Cabannes, chief technology officer at Fresher Information, is the chief architect of the Matisse database, a hybrid database for object developers. For the last 15 years, he's focused on object and database technology, and developing and deploying mission-critical object-based applications in a variety of environments. He holds a master's degree in engineering and has conducted post-graduate research in computer science.

# Compuware Corp.

[www.compuware.com/products/optimalj](http://www.compuware.com/products/optimalj)

# THE CRITICAL ROLE OF APPLICATION ARCHITECTURE



WRITTEN BY WALTER HURST

**A**  
FUNDAMENTAL  
ISSUE FOR  
ENTERPRISE  
APPLICATIONS

**W**HETHER YOU'RE A DEVELOPER WRITING CODE, A MANAGER GUIDING A PROJECT, OR A CUSTOMER GIVING REQUIREMENTS, YOU'RE FAMILIAR WITH THE STEPS NEEDED TO SUCCESSFULLY CREATE A BUSINESS APPLICATION. OFTEN REFERRED TO AS THE APPLICATION-DEVELOPMENT LIFE CYCLE, THESE STEPS TYPICALLY INVOLVE GATHERING REQUIREMENTS, AND DESIGNING, DEVELOPING, TESTING, AND DEPLOYING THE APPLICATION. **SOUND EASY?**

Anyone with any application experience knows that the application development process is fraught with unknowns. In the networked applications space, the unknowns can be even more extreme when it comes to the gap between the technical infrastructure – application servers – and the actual business requirements. More specifically, the architectural issues alone in the “developing the application” stage are often left to the developer or distributed to architectural teams. Several pitfalls can arise in the critical architectural stage:

- **Developers don't know where to start when building an application.**

Often a developer allows the user interface (UI) to drive the business requirements and begins to code. A typical mistake is that the UI developer goes too far when extending the presentation logic with actual business functionality. It then becomes difficult to decouple the functionality and share it across the application, or the developer gets backed into a corner when faced with more difficult issues like data persistence.

- **Developers have difficulty with the steps needed to successfully build an application.**

Even if a development team has been sufficiently trained in the infrastructure technology, such as WebLogic or

Microsoft DCOM, they're still left a blank slate when development begins. This opens up questions like “Where do I start?” “How does all of this technology – EJBs or JMS or DCOM or .NET – help me with what I'm building?” “How do I glue it all together to get to my end application?” Infrastructure technology often leaves the developer with more options than answers.

- **Many developers are solving the same technical problems.**

Take, for example, an e-commerce site. How are decisions such as how to maintain a catalog applied across a project? If the tasks are split up, a developer focused on just one aspect of the application has his or her own deadlines and likely won't have the time or inclination to share this information with others. From a technical perspective, this evolves into more critical decisions, such as how to manage data persistence or validation. For example, “How do I manage input from an HTML form or data type validation?” “What does my validation routine look like?” “What is a valid address?” “How do I call an EJB and what are the steps to do this?” It's easy to see how the development team quickly focuses on nitty-gritty details, not on the application itself. Not only are developers trying to solve the same problems, they're also not sharing these decisions.

# Precise Software

[www.precise.com/jdj](http://www.precise.com/jdj)



- **Project managers don't know what to expect from developers.**

Take, for example, a developer who tells his or her manager that it will take two months to develop the "user flow experience." How does the project manager judge this answer when he or she doesn't have insight into the technical decisions to a working application?

- **Project managers can't easily assign work based on developers' skills, as the delineation of work is often nebulous.**

How do you separate tasks such as business logic from the user interface or from integration? How do you make sure the user interface doesn't extend too far into the business code? This often forces a developer to tackle more issues than he or she needs to, such as addressing security and distribution. Developers start dealing with not just one class but 10 classes equaling thousands of lines of code and often don't have the domain expertise sets to do the work.

- **The application is hard to maintain and extend.**

Applications built without architecture are extremely difficult to update, extend, and modify for basic bug fixes and modifications, as well as for more robust overhauls, to meet changing business requirements. Developers tasked with updating applications spend a majority of their time locating the specific areas they have to change and carefully modifying those areas so as not to break another part of the application.

- **The delineation between the technical details and business logic or functionality of the application is not clear.**

This is most often found when handling persistence in the application, such as how to interact with

oped and its rules.

Sometimes the infrastructure provided by J2EE application servers or the technical plumbing provided by a collection of third-party class libraries is confused with application architecture. Infrastructure components are required for an application, but their role in a distributed application is to provide some "technology feature" such as transaction management, thread management, or distribution. The infrastructure, however, stops short of determining how the application needs to behave.

Application architecture gives development teams the structure for the application and eliminates a significant amount of the coding effort. This structure shields the developer from common development mistakes by providing a better format that eliminates redundancy and inconsistent code and by bringing the actual development closer to the business requirements. Application architecture is reusable across projects allowing:

- Developer resources to be transferable between projects
- Lower costs due to not reinventing the wheel
- Lower maintenance costs due to a uniform approach
- Lower training costs

In the not so distant past, a developer's productivity was measured by how much code he or she wrote. Today, in a world of increasing development complexity and time demands, less is more. Less code means less to create, less to maintain, and less to execute.

#### Application Architecture Approaches: Ignore It, Build It, or ???

In the long history of enterprise applications, there are two answers when approaching application architecture issues:

“

## THE NATURE OF THE BEAST IN APPLICATION DELIVERY IS THAT DEADLINES ARE LOOMING.

THE END RESULT IS THE APPLICATION, NOT THE DOCUMENT, SO TEAMS WILL CUT CORNERS WHERE THEY NEED TO”

the data store. Do you use EJBs to interact with the data store or data access objects? Are you directly calling your EJBs or using Session Facade for this interaction? If your development team's not careful, they've coupled those decisions to the application. Often, trouble is not uncovered until the testing phase. At this point, the cost of overhauling the application is very high.

The bottom line is that the architectural issues behind developing enterprise applications are significant. Without a structure in place, developers are left with undocumented procedures or a verbal design philosophy to guide their development. On top of this, anything ambiguous is left to the team to work out. The nature of the beast in application delivery is that deadlines are looming. The end result is the application, not the document, so teams will cut corners where they need to.

### The Importance of Application Architecture

#### The Role of Application Architecture

Where are the above issues most appropriately addressed? This is the role of application architecture – a very critical step in the application-development process. Unfortunately, this step is often lumped with "develop the application." The application architecture determines how the application is devel-

"don't have one" or "build one." By definition, architecture is the structure of an application. If you have an application, you have an application architecture.

If you chose the "don't have one" approach, what actually occurs is architecture that "just happens." Developers make critical application decisions in isolation or don't apply decisions consistently across the application.

If you chose "build one", whether it's your first, second, or third application, the architectural issues you need to address are significant.

- How are development teams reusing the architecture? How is it being implemented?
- How is it documented? How do I train development teams to use it?
- Is it current? How will you port the architecture as the underlying infrastructure technology changes?
- How do you plan to maintain the current architecture? How do you plan for continually changing resources? How do you take advantage of emerging technologies, such as Web services, if your original resources on the project left?
- What if consultants don't want to work with the current framework because of "spaghetti code" syndrome and are proposing a new approach?



# Canoo Engineering AG

[www.canoo.com/ulc/](http://www.canoo.com/ulc/)

This is just the beginning. Given that development teams are measured on the delivery of the application, not on the “purity” of the architecture, this results in either paper-based or poorly documented implementations that only get used by a single programmer or single project. If problems do arise as a result of the architecture, there’s little chance to correct it. As development teams generally need to move on to the next project, there’s little time allocated to perfecting it. If the application is functioning, the “if it ain’t broke, don’t fix it” mantra rings true when it comes to architecture.

### Problems with Building Architecture

Revisiting where architecture typically fits in the application development life cycle, it’s easy to see why time is a problem. The architecture is usually squeezed between design and development. The customer of the application probably doesn’t care about the architecture and very likely wouldn’t understand it. Because architecture is an intermediate result, quite often it’s not given the proper attention it deserves. The architecture is usually the first thing to give way when there’s a schedule crunch. Typically, the design stage runs over and the final deadline isn’t flexible. Architecture usually suffers first.

The skills required to build a quality architecture are in short supply, making the cost to build it prohibitive. Recent studies show that firms need to budget as much as \$3,500 a day for one architect in the J2EE arena (Forrester Research, “Putting J2EE to Work,” July 2001). One firm claimed that it would take them 12 months just to build the development framework needed to support the application. Architecture marries the business application to the technology infrastructure, and requires the architecture builder to be intimately familiar with application building and the technical infrastructure.

The last problem encountered when building an architecture is the need for iterative development. Consider the evolution of the application development life cycle. Not too long ago, the waterfall approach was very popular. This

### Results of Building – The 3 Ps

The building process results in three types of architecture: a *paper one*, one that a single *programmer* uses, or one that a single *project* uses – paper, programmer, project – the 3Ps.

The paper architecture is actually not an architecture; that is, it isn’t physical code, therefore it leaves the developer at square one when it comes time to build it. Paper architectures usually result when there’s a central think-tank group that determines the overall technical direction. While important architectural decisions are made, the actual code still needs to be written for the architecture.

The programmer architecture involves an individual programmer who builds the application architecture. More than likely other programmers on the same team are solving the same problems in their parts of the application. So you have a single project with little or no consistency across developers.

The project architect is the best possible outcome in a build situation. The need for an architecture has been recognized, but the architecture is likely limited to the project for which it was built. Because the architecture was made to solve the problems of one project, it probably won’t be reusable across other projects. Since the architecture wasn’t a tangible goal of the project, it likely won’t be documented nor will it continue to evolve. In short, it starts and ends with the project.

### Buying Application Architecture

Why do companies historically build architectures? Usually, there’s no alternative. If the infrastructure technology used by a company is unique, then it’s not likely that they would find a commercially available architecture to support the infrastructure. However, with infrastructure standardizing around platforms like J2EE and .NET, buying an application architecture is a viable alternative. It not only eliminates the build challenges discussed earlier, it also provides many benefits such as best practices, future-proofing



“AS COMPANIES BUILD MORE SOPHISTICATED ENTERPRISE APPLICATIONS, PROCEEDING WITHOUT A PROVEN ARCHITECTURE WILL BE FOLLY”

approach broke the work into large chunks (requirements, design, development, etc.) that were done sequentially over a long period of time. The current trend in application development is to use an iterative approach. The steps taken are similar to the old process (e.g., requirements, design, development, etc.) but they’re executed in an iterative fashion over short periods of time. Using small iterations means results are seen sooner and feedback is incorporated more often, resulting in a better product. This same approach is required for a high-quality architecture. Iterations are required to refine the architecture and prove its validity. However, architecture iterations are close to impossible to incorporate into an application development life cycle because of the logistics involved. The developer training, application rewrites, and delivery schedule hinder incorporating architecture iterations.

of the application, significant costs savings, and quicker time-to-market.

### Best Practices

Best practices for application architecture include those for developing code as well as for solving the architectural problems. Becoming familiar with these best practices is done through experience and shared industry expertise. In the world of enterprise applications, an increasingly important source of this industry expertise is design patterns, such as the J2EE Blueprints or object-oriented patterns. Design patterns capture a problem definition and a possible solution in a textual format, allowing people to share their experiences with others to avoid similar problems. Implementing these patterns while staying on top of best practices requires constant attention that only a vendor who is providing the solution can realistically afford.

# SpiritSoft

[www.spiritsoft.com/climber](http://www.spiritsoft.com/climber)



J2ME



J2SE



J2EE



Home

### Future-Proofing Applications

A good application architecture will provide a longer life for your application. Incorporated as a product, the architecture allows the application to easily adopt new functionality and isolates the application from technology changes whether it's upgrading to a new version of a J2EE application server or taking advantage of emerging Web services functionality.

### Less Time and Lower Cost

A very immediate benefit of purchasing an application architecture is lowering the cost of building the application and accelerating the application creation. Many companies are realizing that it's taking 12 months just to get their development framework or application architecture implemented. That's not even taking into account the actual construction of the business application itself.

A good application architecture product needs:

- To be tested, proven, and built on industry best practices
- To have extensive documentation
- To be supported by a company that's focused on providing architecture and that will continue to maintain and extend the offering
- To provide formal training to get the application team up to speed quickly

Just as in the early days of object-oriented, Java, and Web application development, many companies attempted to build their own distributed services (load balancing, security, dynamic page generation, and scripting) to support their applications. Tackling this in-house was cost-prohibitive, in terms of time-to-market and maintenance, and opened a door for an application server market in which lead vendors assumed the role of providing this infrastructure (BEA, IBM, and others). Can you imagine having a conversation with your CIO about wanting to build your own application server?

As companies build more sophisticated enterprise applications, proceeding without a proven architecture will be folly. How you approach application architecture and address its considerations across the application, as well as across future applications, is increasingly critical to your overall success.

### Conclusion

Application architecture is critical to the initial success of an application and the ongoing success of maintaining and extending it. Companies have historically built their own development frameworks with varying degrees of success. With no other options available, development teams were left to poorly documented, unrepeatable application architectures.

As standard technical infrastructures such as J2EE and .NET continue to mature over time, development teams will gather more enterprise experience grappling with how best to divide up work tasks, utilize the infrastructure technology, and deliver applications on time. With growing industry acceptance of these infrastructures, vendors are delivering tools and frameworks to ease the next level of application development for many project teams, allowing them to focus on the business requirements at hand and obviating the need to deal with what will soon become mundane development tasks. Nowhere will this be more apparent than in the area of application architecture. ●

#### AUTHOR BIO

Walter Hurst is the CTO and cofounder of Wakesoft, a provider of prebuilt application architecture and frameworks for J2EE. He received a BS in computer engineering from the University of Michigan.

[whurst@wakesoft.com](mailto:whurst@wakesoft.com)

## SYS-CON MEDIA

PUBLISHER, PRESIDENT, AND CEO  
FUAT A. KIRCAALI [fuat@sys-con.com](mailto:fuat@sys-con.com)  
VICE PRESIDENT, BUSINESS DEVELOPMENT  
GRISHA DAVIDA [grisha@sys-con.com](mailto:grisha@sys-con.com)  
CHIEF OPERATING OFFICER  
MARK HARABEDIAN [mark@sys-con.com](mailto:mark@sys-con.com)

### ADVERTISING

SENIOR VICE PRESIDENT, SALES AND MARKETING  
CARMEN GONZALEZ [carmen@sys-con.com](mailto:carmen@sys-con.com)  
VICE PRESIDENT, SALES AND MARKETING  
MILES SILVERMAN [miles@sys-con.com](mailto:miles@sys-con.com)  
ADVERTISING SALES DIRECTOR  
ROBYN FORMA [robyn@sys-con.com](mailto:robyn@sys-con.com)  
ADVERTISING ACCOUNT MANAGER  
MEGAN RING [megan@sys-con.com](mailto:megan@sys-con.com)  
ASSOCIATE SALES MANAGERS  
CARRIE GEBERT [carrie@sys-con.com](mailto:carrie@sys-con.com)  
KRISTIN KUHNLE [kristen@sys-con.com](mailto:kristen@sys-con.com)  
ALISA CATALANO [alisa@sys-con.com](mailto:alisa@sys-con.com)  
LEAH HITTMAN [leah@sys-con.com](mailto:leah@sys-con.com)

### EDITORIAL

EXECUTIVE EDITOR  
NANCY VALENTINE [nancy@sys-con.com](mailto:nancy@sys-con.com)  
EDITOR  
M'LOU PINKHAM [mpinkham@sys-con.com](mailto:mpinkham@sys-con.com)  
MANAGING EDITOR  
CHERYL VAN SISE [cheryl@sys-con.com](mailto:cheryl@sys-con.com)  
ASSOCIATE EDITORS

JAMIE MATUSOW [jamie@sys-con.com](mailto:jamie@sys-con.com)  
GAIL SCHULTZ [gail@sys-con.com](mailto:gail@sys-con.com)  
JEAN CASSIDY [jean@sys-con.com](mailto:jean@sys-con.com)  
ONLINE EDITOR

LIN GOETZ [lin@sys-con.com](mailto:lin@sys-con.com)  
PRODUCTION  
VICE PRESIDENT, PRODUCTION AND DESIGN  
JIM MORGAN [jim@sys-con.com](mailto:jim@sys-con.com)  
LEAD DESIGNER

LOUIS F. CUFFARI [louis@sys-con.com](mailto:louis@sys-con.com)  
ART DIRECTOR  
ALEX BOTERO [alex@sys-con.com](mailto:alex@sys-con.com)  
ASSOCIATE ART DIRECTORS  
CATHRYN BURAK [cathyb@sys-con.com](mailto:cathyb@sys-con.com)  
RICHARD SILVERBERG [richards@sys-con.com](mailto:richards@sys-con.com)  
AARATHI VENKATARAMAN [aarathi@sys-con.com](mailto:aarathi@sys-con.com)  
ASSISTANT ART DIRECTOR  
TAMI BEATTY [tami@sys-con.com](mailto:tami@sys-con.com)

### WEB SERVICES

WEBMASTER  
ROBERT DIAMOND [robert@sys-con.com](mailto:robert@sys-con.com)  
WEB DESIGNERS  
STEPHEN KILMURRAY [stephen@sys-con.com](mailto:stephen@sys-con.com)  
CHRISTOPHER CROCE [chris@sys-con.com](mailto:chris@sys-con.com)

### ACCOUNTING

CHIEF FINANCIAL OFFICER  
BRUCE KANNER [bruce@sys-con.com](mailto:bruce@sys-con.com)  
ASSISTANT CONTROLLER  
JUDITH CALNAN [judith@sys-con.com](mailto:judith@sys-con.com)  
ACCOUNTS RECEIVABLE  
JAN BRAIDECH [jan@sys-con.com](mailto:jan@sys-con.com)  
ACCOUNTS PAYABLE  
JOAN LAROSE [joan@sys-con.com](mailto:joan@sys-con.com)  
ACCOUNTING CLERK  
BETTY WHITE [betty@sys-con.com](mailto:betty@sys-con.com)

### SYS-CON EVENTS

VICE PRESIDENT, SYS-CON EVENTS  
CATHY WALTERS [cathyw@sys-con.com](mailto:cathyw@sys-con.com)  
CONFERENCE MANAGER  
MICHAEL LYNCH [mike@sys-con.com](mailto:mike@sys-con.com)  
SALES EXECUTIVES, EXHIBITS  
MICHAEL PESICK [michael@sys-con.com](mailto:michael@sys-con.com)  
RICHARD ANDERSON [richard@sys-con.com](mailto:richard@sys-con.com)

### CUSTOMER RELATIONS/JDJ STORE

MANAGER, CUSTOMER RELATIONS/JDJ STORE  
ANTHONY D. SPITZER [tony@sys-con.com](mailto:tony@sys-con.com)

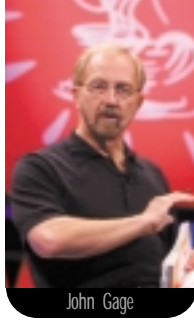
# SilverStream Software

[www.silverstream.com/coals](http://www.silverstream.com/coals)

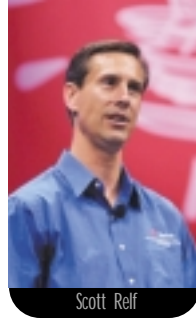




Jim Balsillie



John Gage



Scott Relf



Scott McNealy



James Gosling



Amit Pau

# First Impressions

WRITTEN BY AJIT SAGAR

## JavaOne™ 2002 Show Review

Sun's seventh annual JavaOne conference was held March 25–28 at the Moscone Center in downtown San Francisco. This was the sixth Java conference I attended and it was interesting to compare it with the previous shows.

### *It's All About Networking*

JavaOne, as always, is a great conference for networking. And I don't mean the networking that requires wires for connectivity. I'm referring to people hooking up with others who are using the Java platform to solve business problems. This JavaOne was no different. I chatted with several attendees about what they expected from the conference, and what they got out of it.

The majority felt the conference was a great way to generate leads and to get an idea of who else was leveraging Java and how; shared experiences, war stories, and potential alliances were some of the outcomes. Many of the attendees who had been at previous JavaOnes felt the sessions were not as exciting as in



the past. However, the majority felt that the exhibition hall and the vendor demos were very interesting. Many felt that by attending the conference, they had generated leads they normally wouldn't have. On the other hand, the information in the sessions could have easily been picked up on the Internet. Most attendees felt that their trip to JavaOne gave them a better understanding of what's available in the market to build real-world applications.

Based on estimates at Sun's site, there were:

- Over 50 countries represented
- Over 350 members of the press
- 35 cosponsors
- 200 exhibitors

- 10 media cosponsors
- 300 in-depth technical sessions
- 200 birds-of-a-feather sessions

### *Attendance*

My first impression was that attendance was down, and the atmosphere was more sober than in previous years. The obvious reason for this was that companies have cut back on spending and it's hard to justify using already-stretched resources. Another thing of note was that international representation was not as high as in previous years. Again, with concerns over safety and travel expenses, this wasn't unexpected.

In addition, the conference took place during the week of Passover, ruling out attendance from the Jewish community. This year's attendance seemed dominated by the Bay area folks and seemed to consist of seasoned developers who were involved in the development of real-world applications. I didn't come across many newbies in the crowd.



# Altova

[www.altova.com](http://www.altova.com)



Ernie Cormier



Rob Gingell



Alfred Chuang



Hasso Plattner



Rich Green



Bill Boggess



### The Message

The message was clear. Sun's new logo "We make the Net work" seems a lot more palatable than "The network is the computer." The Java platform is coming of age and it's all about enabling the development of distributed applications. The maturity of the products displayed in the exhibition hall was quite apparent. The main additions to the platform consisted mostly of the enabling APIs for developing Web services. Web services, wireless connecti-

ty, and enterprise applications based on J2EE were the cornerstones of the conference.

### Keynotes

Besides Sun's keynote presentations on the vision of Java technology and Web services, additional themes centered around the role of Java in wireless and small devices, enterprise integration, and telecommunications. Unlike last year, when the keynotes focused on Web services - then the brand-new fad

- I felt they were more balanced this year. Monday's presentation indicated that Sun was supporting Web services initiatives, and the Java platform has ample specifications and tools coming down the pipe to support building distributed applications based on Web services.

### Technology Showcase

The Sharp Zaurus PDA, bundled with the Linksys 802.11b wireless network card, was the official device promoted this year. It was available to attendees at a special show price of \$299, way below market price. The Technology Showcase featured PersonalJava technology, SOAP, Web services, JXTA, JXTA for J2ME, and an 802.11b wireless network. Developers used this device to work through the competitions hosted at the conference - "What Time Is It?" and the "Hackathon."



# INT, Inc

[www.int.com](http://www.int.com)



# DataDirect Technologies

[www.datadirect-technologies.com](http://www.datadirect-technologies.com)



Paul Saffo



Blake Stone



Patricia Suelz



Jason Hunter



Jeff Jackson



Ivo Toter



### Session Themes

As mentioned earlier, Web services was one of the themes of the conference, but the content of the sessions was much more balanced. I attended some of the sessions under the “Java Technologies, Products, Solutions, and You” track and was quite happy to find that others in the industry have faced similar victories and defeats with Java technologies. Overall, the real-world examples of applications were reassuring, as they indicated the commitment Java has from leading companies around the world.

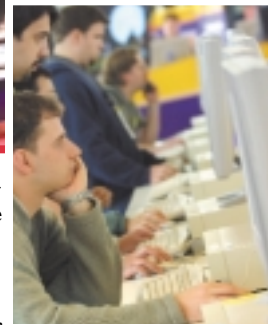
There were few sessions on new APIs for the simple reason that the Java platform hasn't added major modules to the framework. This is a good thing, as it indicates a maturing of the platform. Sessions that introduced new modules centered around Java 1.4, EJB2.1, the Java XML Service Pack, and the Sun ONE Web Services initiative. There were also several interesting sessions on Java



design patterns, architecture, and language usage.

### Pavilion

The pavilion was, for me, the most interesting part of the show. I met with most of the vendors on the exhibit floor and was very impressed by the products coming out in the Java market. They are less experi-



mental and more geared toward supporting serious developers working on serious applications. The Java tools market has matured a lot in the past couple of years. This was evident by the fact that most of the IDE vendors were exhibiting tools that addressed integrated application development (instead of stand-alone), enterprise application testing,

# Capella University

[www.capellauniversity.edu](http://www.capellauniversity.edu)

“This year's **JavaOne** was a testimony to how far Java has come toward becoming a mature and robust platform

debugging, and logging. The design tools offer environments that allow developers and business process modelers to work at more abstract levels. The application server vendors who showed up at the conference are the ones who have emerged with solid products in the Java market.

The vendor booths were more spread out in the exhibit hall, primarily because there were fewer vendors than in previous years. However, there was traffic throughout the day, which indicated that most of the attendees were seriously interested in what the vendors had to offer. It also seems like marketing budgets have been replenished, as there were more toys, T-shirts, etc., handed out.

### SYS-CON Radio

At **SYS-CON's** popular radio booth, leading vendors and industry luminaries were interviewed throughout the day. I interviewed several of the vendor representatives and, overall, the mood was enthusiastic and upbeat.

All in all, this year's JavaOne was a testimony to how far Java has come toward becoming a mature and robust platform that's widely adopted by the computing industry. ●

# AltoWeb

[www.altoweb.com](http://www.altoweb.com)



KEITH BROWN J2SE EDITOR

## We've Built the House

If you read my editorial last month (*JDJ*, Vol. 7, issue 4), you'll recall that I was trying to work out just who the Java community was and whether or not you or I feel a part of it. Well, I think I met the community at JavaOne 2002.

During JavaOne we spoke to over 100 people at **SYS-CON Radio**, and recorded and uploaded these conversations to *JDJ*'s Web site. Some are more interesting than others, and some have more to offer the Java developer.

There's a lot of jargon to wade through in these interviews and it's often tricky to identify what it all means.

Two years ago, the big thing that so many companies were talking about was *application servers* and how they were going to save the world. That area of the Java world has now consolidated and the emphasis at JavaOne had progressed to the next stage – software to analyze our software!

There was a major focus on testing and performance tools. Java technology has matured to a point where applications and systems are now so widespread that the market is demanding tools to fine-tune the software's efficiency and performance.

We've built the house. Now we need to find out where those drafts are coming from and fit some double-glazing.

Besides the software-tuning vendors, it's always interesting to speak to authors and discuss their specific topics. As authors, they tend to be effective communicators and speak in a language developers are used to.

Adam Kolawa spoke to us animatedly about his new book, *Bulletproofing Web Applications* (coauthored by Kolawa,

Wendell Hicken, and Cynthia Dunlop and published by Hungry Minds, Inc.), and how we, as developers, can ensure our Web applications are reliable and consistent. The book contains an interesting section on JSP, and the interview is well worth a listen ([www.sys-con.com/java/javaone2002b.cfm](http://www.sys-con.com/java/javaone2002b.cfm)).

Iain Shiegoka also gave a good overview of Java instant messaging and explains what it's like to write these types of books. His book, *Instant Messaging in Java* (Manning Publications), concentrates on the open source Jabber XML-based IM protocols ([www.sys-con.com/java/javaone2002c.cfm](http://www.sys-con.com/java/javaone2002c.cfm)).

It was also nice to see a few familiar faces at the **SYS-CON Radio** booth, including Ralf Dossmann from Borland, the first person I ever interviewed; that first interview was two years ago at the IBM Java Conference in Austria. Further evidence that the Java community not only exists, but is also persistent!

In addition to talking with exhibitors and attendees at the conference, there's also a lot of socializing, a little gossip, and a bit of eavesdropping – it can't be helped.

One interesting discussion I overheard was about Swing's JTree component. The basic gist of the debate – that I might, slightly tongue in cheek, agree with – was that JTree should have a certification program all its own. Perhaps a Sun Certified JTree Developer badge will be my next goal. Anyone who has wrestled with the JTree and come out alive will know that it's not a particularly easy task to produce the tree you want. I was glad to hear others had the same experience. You're not alone! ☘

keith.brown@sys-con.com

### AUTHOR BIO

Keith Brown has been involved with Java for many years. When he's not coding up client solutions for a European Java company, he can be found lurking in the corridors of conferences all around the world.

### We've Built the House

If you read my editorial last month (*JDJ*, Vol. 7, issue 4), you'll recall that I was trying to work out just who the Java community was and whether or not you or I feel a part of it. Well, I think I met the community at JavaOne 2002.

by Keith Brown

36

### Programming Neural Networks in Java

How to construct a simple, yet particle, neural network in Java that can recognize handwritten letters, and implement a neural network in a small sample program.

by Jeff Heaton

38

### Better Scaling with New I/O

With J2SE version 1.4, Java finally has a scalable I/O API. This article shows how to write a simple Web server with both the new and the old API.

by Hendrik Schreiber

46

### JDBC 3.0 – Something for Everyone

There was no ticker tape parade to accompany the release of the JDBC 3.0 specification, but many will be pleasantly surprised at its list of enhancements that include everything from performance-tuning options to support for extended-level database features.

by John Goodson

56

# Addison-Wesley

# Programming Neural Networks in Java

An efficient way to perform certain operations



WRITTEN BY  
JEFF HEATON

Computers can perform many operations a lot faster than humans. However, there are many tasks in which the computer falls considerably short. One such task is the interpretation of graphic information. A preschool child can easily tell the difference between a cat and a dog, but this simple problem confounds today's computers.

In the field of computer science, artificial intelligence attempts to give computers human abilities. One of the primary means by which computers are endowed with humanlike abilities is through the use of a neural network, which the human brain is the ultimate example of. The human brain consists of a network of over a billion interconnected neurons. These are individual cells that can process small amounts of information and then activate other neurons to continue the process. However, the term *neural network*, as it's normally used, is actually a misnomer. Computers attempt to simulate a neural network. However, most publications use the term *neural network* rather than *artificial neural network*.

This article shows how to construct a neural network in Java; however, they can be constructed in almost any programming language. Most publications about neural networks use such computer languages as C, C++, Lisp, or Prolog. Java is actually quite effective as a neural network programming language. This article shows you a simple, yet particle, neural network that can recognize handwritten letters, and describes the implementation of a neural network in a small sample program. (All sample programs and source code for this article can be downloaded from [www.sys-con.com/java/sourcec.cfm](http://www.sys-con.com/java/sourcec.cfm).)

## Recognizing Letters

Using the sample program (shown in Figure 1) you can see a neural network in action. For ease of distribution, the class files are packaged into a single JAR file named `OCR.jar`. To run the program, use the following command (assuming you're in the same directory as the JAR file). Some systems may allow you to simply double-click the JAR file.

```
java -classpath OCR.jar MainEntry
```

When the letter-recognition program begins, there's no data loaded initially. A training file must be used that contains the shapes of the letters. An example training file (`sample.dat`) is preloaded with the 26 capital letters. To see the program work, click the "Load" button, which loads the `sample.dat` file. Now 26 letter patterns are in memory and the network must be trained. Click the "Begin Training" button; now the network is ready to recognize characters. Draw any capital letter you like and click "Recognize"; the program should now recognize the letter.

## Training the Sample Program

Maybe my handwriting is considerably different than most people's. (My first grade teacher would certainly say so.) What if you want to train the program specifically for your handwriting? To replace a letter that's already defined, you must select and delete that letter first. Pressing the "Delete" button does this. Now draw the character you wish to train the program for. If you'd like to see this letter downsampled before you add it, click the "Downsample" button. If you're happy with your letter, click the "Add" button to add it to the training set. To save a copy of your newly created letters, click the "Save" button and they'll be written to the `Sample.dat` file.

Once you've entered all the letters you want, you must now "train" the neural network. Up to this point you've simply provided a training set of letters known as *input patterns*. With these input patterns, you're now ready to train the network, which could take a lot of time. However, since only one drawing sample per letter is allowed, this process

will be completed in a matter of seconds. A small popup will be displayed when training is complete. When you save, only the character patterns are saved. If you load these same patterns later, you must retrain the network.

You'll now be shown how this example program is constructed, and how you can create similar programs. The file `MainEntry.java` contains the Swing application that makes up this application, which is little more than placing the components at their correct locations.

The three areas this article focuses on are downsampling, training, and recognition. Downsampling, an algorithm used to reduce the resolution of the letters being drawn, is used for character recognition and training, so we'll address this topic first.

## Downsampling the Image

All images are downsampled before being used, which prevents the neural network from being confused by size and position. The drawing area is large enough so you could draw a letter in several different sizes. By downsampling the image to a consistent size, it won't matter how large you draw the letter, as the downsampled image will always remain a consistent size. This section shows how this is done.

When you draw an image, the program first draws a box around the boundary of your letter. This allows the program to eliminate all the white space around your letter. This process is done inside the "downsample" method of the `Entry.java` class. As you draw a character, this character is also drawn onto the "entryImage" instance variable of the `Entry` object. To crop this image and eventually downsample it, we must grab the bit pattern of the image. This is done using a `PixelGrabber` class:

# ESRI

[www.esri.com/mapobjectsjava](http://www.esri.com/mapobjectsjava)



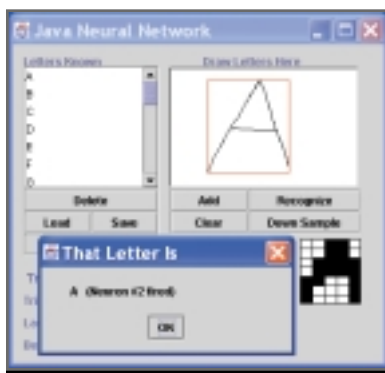


FIGURE 1 The character recognition program

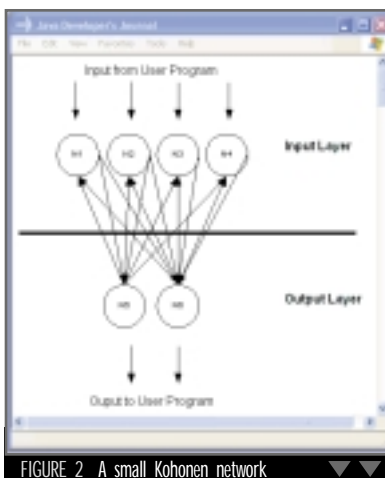


FIGURE 2 A small Kohonen network

```
int w = entryImage.getWidth(this);
int h = entryImage.getHeight(this);

PixelGrabber grabber = new
PixelGrabber(entryImage,

0,0,w,h,true);
grabber.grabPixels();
pixelMap = (int[])grabber.getPixels();
```

After this code completes, the `pixelMap` variable, which is an array of `int` datatypes, now contains the bit pattern of the image. The next step is to crop the image and remove any white space. Cropping is implemented by dragging four imaginary lines from the top, left, bottom, and right sides of the image. These lines will stop as soon as they cross an actual pixel. By doing this, these lines snap to the outer edges of the image. The `hLineClear` and `vLineClear` methods both accept a parameter that indicates the line to scan, and returns `true` if that line is clear. The program works by calling `hLineClear` and `vLineClear` until they cross the outer edges of the image. The horizontal line method (`hLineClear`) is shown here.

```
protected boolean hLineClear(int y)
{
    int w = entryImage.getWidth(this);
```

```
for ( int i=0;i<w;i++) {
    if ( pixelMap[(y*w)+i] !=-1 )
        return false;
    }
return true;
}
```

As you can see, the horizontal line method accepts a `y` coordinate that specifies the horizontal line to check. The program then loops through each `x` coordinate on that row, checking for any pixel values. The value of `-1` indicates white, so it's ignored. The "findBounds" method uses "hLineClear" and "vLineClear" to calculate the four edges. The beginning of this method is shown here:

```
protected void findBounds(int w,int h)
{
    // top line
    for ( int y=0;y<h;y++) {
        if ( !hLineClear(y) ) {
            downSampleTop=y;
            break;
        }
    }
    // bottom line
    for ( int y=h-1;y>=0;y-- ) {
        if ( !hLineClear(y) ) {
            downSampleBottom=y;
            break;
        }
    }
}
```

You can see how the program calculates the top and bottom lines of the cropping rectangle. To calculate the top line, the program starts at 0 and continues to the bottom of the image. As soon as the first nonclear line is found, the program establishes this as the top of the clipping rectangle. The same process, only in reverse, is carried out to determine the bottom of the image. The processes to determine the left and right boundaries are carried out in the same way.

Now that the image has been cropped, it must be downsampled. This involves taking the image from a larger resolution to a 5x7 resolution. To reduce an image to 5x7, think of an imaginary grid being drawn over the high-resolution image. This divides the image into rectangular sections, five across and seven down. If any pixel in a section is filled, the corresponding pixel in the 5x7 downsampled image is also filled. Most of the work done by this process is accomplished inside the "downSampleQuadrant" method shown here.

```
protected boolean down-
SampleQuadrant(int x,int y)
```

```
{
    int w =entryImage.getWidth(this);
    int startX =(int)
        (downSampleLeft+(x*ratioX));
    int startY = (int)
        (downSampleTop+(y*ratioY));
    int endX = (int)(startX +
        ratioX);
    int endY = (int)(startY +
        ratioY);

    for ( int yy=startY;yy<=endY;yy++
        ) {
        for ( int xx=startX
            ;xx<=endX;xx++ ) {
            int loc = xx+(yy*w);

            if ( pixelMap[ loc ] != -1 )
                return true;
        }
    }

    return false;
}
```

The "downSampleQuadrant" method accepts the section number that should be calculated. First the starting and ending `x` and `y` coordinates must be calculated. To calculate the first `x` coordinate for the specified section, first the "downSampleLeft" is used; this is the left side of the cropping rectangle. Then `x` is multiplied by "ratioX", the ratio of how many pixels make up each section. This allows us to determine where to place "startX". The starting `y` position, "startY", is calculated by similar means. Next the program loops through every `x` and `y` covered by the specified section. If even one pixel is determined to be filled, the method returns `true`, which indicates that this section should be considered filled.

The "downSampleQuadrant" method is called in succession for each section in the image. This results of the sample image are stored in the "SampleData" class, a wrapper class that contains a 5x7 array of Boolean values. It's this structure that forms the input to both training and character recognition.

### Neural Network Recognition

There are many types of neural networks, and most are named after their creators. I'll be using a Kohonen neural network, a two-level network (see Figure 2). The downsampled character pattern drawn by the user is fed to the input neurons. There's one input neuron for every pixel in the downsampled image. Because the downsampled image is a 5x7 grid, there are 35 input neurons.

Through the output neurons, the neural network communicates which



# Dice

[www.dice.com](http://www.dice.com)

letter it thinks the user drew. The number of output neurons always matches the number of unique letter samples that were provided. Since 26 letters were provided in the sample, there will be 26 output neurons. If this program were modified to support multiple samples per letter, there would still be 26 output neurons, even if there were multiple samples per letter.

In addition to input and output neurons, there are also connections between the individual neurons. These connections are not all equal. Each is assigned a weight, which is ultimately the only factor that determines what the network will output for a given input pattern. To determine the total number of connections, multiply the number of input neurons by the number of output neurons. A neural network with 26 output neurons and 35 input neurons would have a total of 910 connection weights. The training process is dedicated to finding the correct values for these weights.

The recognition process begins when the user draws a character and then clicks the “Recognize” button. First the letter is downsampled to a 5x7 image. This image must be copied from its two-dimensional array to an array of doubles that will be fed to the input neurons.

```
entry.downSample();

double input[] = new double[5*7];
int idx=0;
SampleData ds = sample.getData();
for ( int y=0;y<ds.getHeight();y++ )
    {
        for ( int x=0;x<ds.getWidth();x++ )
            {
                input[idx++] =
                    ds.getData(x,y)?.5:-.5;
            }
    }
}
```

This code does the conversion. Neurons require floating point input. As a result, the program feeds it the value of 5 for a white pixel and -5 for a black pixel. This array of 35 values is fed to the

input neurons by passing the input array to the Kohonen’s “winner” method. This returns which of the 35 neurons won and is stored in the “best” integer.

```
int best = net.winner ( input , norm-
    fac , synth ) ;
char map[] = mapNeurons();
```

```
JOptionPane.showMessageDialog(this,
    " " + map[best] + " (Neuron #"
    + best + " fired)","That Letter Is",
    JOptionPane.PLAIN_MESSAGE);
```

Knowing the winning neuron is not too helpful because it doesn’t show you which letter was recognized. To line up the neurons with their recognized letters, each letter image the network was trained from must be fed into the network and the winning neuron determined. For example, if you were to feed the training image for “J” into the neural network, and the winning neuron were neuron #4, you would know that it’s the one that had learned to recognize J’s pattern. This is done by calling the “mapNeurons” method, which returns an array of characters. The index of each array element corresponds to the neuron number that recognizes that character.

Most of the actual work performed by the neural network is done in the winner method. The first thing the winner method does is normalize the inputs and calculate the output values of each output neuron. The output neuron with the largest output value is considered the winner. First the “biggest” variable is set to a very small number to indicate there’s no winner yet.

```
biggest = -1.E30;
for ( i=0 ; i<outputNeuronCount;
    i++ ) {
    optr = outputWeights[i];
    output[i] = dotProduct (input ,
        optr ) * normfac[0]
        + synth[0] *
        optr[inputNeuronCount] ;
    // Remap to bipolar(-1,1 to
    0,1)
```

```
output[i] = 0.5 * (output[i] +
    1.0) ;
if ( output[i] > biggest ) {
    biggest = output[i] ;
    win = i ;
}
```

Each output neuron’s weight is calculated by taking the dot product of each output neuron’s weights to the input neurons. The dot product is calculated by multiplying each of the input neuron’s input values against the weights between that input neuron and the output neuron. These weights were determined during training, which is discussed in the next section. The output is kept, and if it’s the largest output so far, it’s set as the “winning” neuron.

As you can see, getting the results from a neural network is a quick process. Actually determining the weights of the neurons is the complex portion of this process. Training the neural network is discussed in the following section.

### How the Neural Network Learns

Learning is the process of selecting a neuron weight matrix that will correctly recognize input patterns. A Kohonen neural network learns by constantly evaluating and optimizing a weight matrix. To do this, a starting weight matrix must be determined. This matrix is chosen by selecting random numbers. Of course, this is a terrible choice for a weight matrix, but it gives a starting point to optimize from.

Once the initial random weight matrix is created, the training can begin. First the weight matrix is evaluated to determine what its current error level is. This error is determined by how well the training input (the letters that you created) maps to the output neurons. The error is calculated by the “evaluateErrors” method of the KohonenNetwork class. If the error level is low, say below 10%, the process is complete.

When the user clicks the “Begin Training” button, the training process begins with the following code:

```
int inputNeuron = MainEntry.DOWN
    SAMPLE_HEIGHT*
    MainEntry.DOWNSAMPLE_WIDTH;
int outputNeuron = letter
    ListModel.size();
```

This calculates the number of input and output neurons. First, the number of input neurons is determined from the size of the downsampled image. Since the height is 7 and the width is 5, the number of input neurons will be 35. The number of output neurons matches the

Learning is the process of selecting a neuron weight matrix that will correctly recognize input patterns

# Rational Software

[www.rational.com/offer/javacd2](http://www.rational.com/offer/javacd2)

## Neural networks should be considered anytime complex patterns must be recognized

number of characters the program has been given.

This is the part of the program that could be modified if you want it to accept and train from more than one sample per letter. For example, if you wanted to accept four samples per letter, you'd have to make sure that the output neuron count remained 26, even though 104 input samples were provided to train with (4 for each of the 26 letters).

Now that the size of the neural network has been determined, the training set and neural network must be constructed. The training set is constructed to hold the correct number of "samples." This will be the 26 letters provided.

```
TrainingSet set = new TrainingSet(inputNeuron,outputNeuron);
set.setTrainingSetCount(letterListModel.size());
```

Next, the downsampled input images are copied to the training set; this is repeated for all 26 input patterns.

```
for ( intt=0;t<letterListModel.size();t++) {
    int idx=0;
    SampleData ds = (SampleData)
        letterListModel.getElementAt(t);
    for ( int y=0;y<ds.getHeight();y++ ) {
        for ( int x=0;x<ds.getWidth();x++ ) {
            set.setInput(t,idx++,ds.getData(x,y)?.
                5:-.5);
        }
    }
}
```

Finally the neural network is constructed and the training set is assigned, so the "learn" method can be called. This will adjust the weight matrix until the network is trained.

```
net = new KohonenNetwork(inputNeuron,outputNeuron,this);
net.setTrainingSet(set);
net.learn();
```

The learn method will loop up to an unspecified number of iterations. Because this program only has one sample per output neuron, it's unlikely that it will take more than one iteration. When the number of training samples matches the output neuron count, training occurs very quickly.

```
n_retry = 0 ;
for ( iter=0 ; ; iter++ ) {
```

A method, "evaluateErrors", is called to evaluate how well the current weights are working. This is determined by looking at how well the training data spreads across the output neurons. If many output neurons are activated for the same training pattern, then the weight set is not a good one. An error rate is calculated, based on how well the training sets are spreading across the output neurons.

```
evaluateErrors ( rate , learnMethod , won , bigerr , correc , work ) ;
```

Once the error is determined, we must see if it is below the best error we've seen so far. If it is, this error is copied to the best error, and the neuron weights are also preserved.

```
totalError = bigerr[0] ;
if ( totalError < best_err ) {
    best_err = totalError ;
    copyWeights ( bestnet , this ) ;
}
```

The total number of winning neurons is then calculated, allowing us to determine if no output neurons were activated. In addition, if the error is below the accepted quit error (10%), the training stops.

```
winners = 0 ;
for ( i=0;i<won.length;i++ )
    if ( won[i]!=0 )
        winners++;
```

```
if ( bigerr[0] < quitError )
    break ;
```

If there is not an acceptable number of winners, one neuron is forced to win.

```
if ( (winners < outputNeuronCount) &&
    (winners < train.getTrainingSetCount()) ) {
    forceWin ( won ) ;
    continue ;
}
```

Now that the first weight matrix has been evaluated, it's adjusted based on its error. The adjustment is slight, based on the correction that was calculated when the error was determined. This two-step process of adjusting the error calculation and adjusting the weight matrix is continued until the error falls below 10%.

```
adjustWeights ( rate , learnMethod , won , bigcorr , correc ) ;
```

This is the process by which a neural network is trained. The method for adjusting the weights and calculating the error is shown in the KohonenNetwork.java file.

### Conclusion

The example presented here is very modular. The neural network Java files contained in this example are KohonenNetwork.java, Network.java, and TrainingSet.java. These files do not pertain to character recognition. The other files are responsible for the user interface and downsampling. One limitation mentioned in the article is that only one drawing can be defined per character. The underlying Kohonen network classes would easily support this feature. This is something that could be added to the user interface with a few more classes.

Neural networks provide an efficient way of performing certain operations that would otherwise be very difficult. Consider how a character recognition program would work without neural networks. You'd likely find yourself writing complex routines that traced outlines, analyzed angles, and did other graphical analysis. Neural networks should be considered anytime complex patterns must be recognized. These patterns don't need to be graphical in nature. Any form of data that can have pattern is a candidate for a neural network solution. ☛

heatonj@heat-on.com

### AUTHOR BIO

Jeff Heaton, a software designer for the Reinsurance Group of America (RGA), is a member of the IEEE and a Sun-certified Java programmer. He's the author of Programming Spiders, Bots, and Aggregators in Java.

# Prentice Hall PTR

[www.newatlanta.com](http://www.newatlanta.com)

# BETTER SCALING WITH NEW

# I/O

THE END OF THE THREAD/SOCKET MARRIAGE

WRITTEN BY HENDRIK SCHREIBER

# W

WITH  
J2SE  
VERSION  
1.4, JAVA  
FINALLY HAS  
A SCALABLE I/O  
API. NOT THAT  
THE OLD API WAS  
AN ABSOLUTE  
FAILURE (JAVA'S  
TREMENDOUS SUCCESS  
IN THE APPLICATION  
SERVER MARKET REFUTES  
THIS), BUT SOME OF THE OLD  
API'S PROPERTIES LED TO DRASTIC  
RESTRICTIONS. THE WORST ONE  
WAS THE BLOCKING I/O.



J2ME



J2SE




J2EE



Home





To write data over a socket, you have to call the `write()` method of an associated `OutputStream`. This call returns only after you've written all the necessary bytes. Given that the send buffers are full and the connection is slow, this might take a while. If your program operates only with a single thread, other connections have to wait, even if they're ready to process `write()` calls. To work around this problem, you have to associate a thread with each socket. This way one thread can work while another one is blocked due to I/O-related tasks.

Threads aren't as heavyweight as real processes. But, depending on the underlying platform, they're not resource savers either. Each thread uses a certain amount of memory and, apart from that, many threads imply many thread-context switches, which aren't cheap.

Java needed a new API to separate the all-too-happy marriage of socket and thread. This finally happened with the new I/O API (`java.nio.*`).

In this article I show how you can write a simple Web server with both the new and the old API. Since HTTP, the Web's protocol, is not as trivial as it used to be, I'll realize only some simple central features. Therefore, the programs shown here are neither secure nor protocol-conforming.

### Old School Httpd

Let's look at the old-school HTTP server first (see Listing 1). (Listings 1-5 can be downloaded from [www.sys-con.com/java/sourcec.cfm](http://www.sys-con.com/java/sourcec.cfm).) Since I need only a single class for this realization, it's quickly explained. In the `main()` method, a `ServerSocket` is instantiated and bound to port 8080. Of course, you'd usually bind a Web server to port 80, but on Unix systems you can only do this with superuser rights. Fortunately, not everyone has them, which is why I chose to use port 8080.

```
public static void main() throws IOException {
    ServerSocket serverSocket = new ServerSocket(8080);
    for (int i=0; i < Integer.parseInt(args[0]); i++) {
        new Httpd(serverSocket);
    }
}
```

Then a number of `Httpd` objects are created and initialized with the shared `ServerSocket`. In the `Httpd`'s constructor, I make sure all instances have a meaningful name, set a default protocol, and start the server by executing the `start()` method of its superclass `Thread`. This leads to an asynchronous call to the `run()` method, in which an infinite loop is located.

In this infinite loop, the `ServerSocket`'s blocking `accept()` method is called. When a client connects to port 8080 of the server, the `accept()` method will return a socket object. Associated with each socket are an `Input`- and an `OutputStream`. Both are used in the following call to the `handleRequest()` method. In this method the client's request is read, checked, and an appropriate response is sent back. If it's a legitimate request, the requested file is sent back using `sendFile()`. If it's not, the client will receive a corresponding error message (`sendError()`). To keep things simple, I won't discuss the specifics of the protocol.

```
while (true) {
    ...
    socket = serverSocket.accept();
    ...
    handleRequest();
    ...
    socket.close();
}
```





# N

ow let's think about this realization for a second. Does it perform well? On the whole, yes. Certainly I could optimize the request parsing – the StringTokenizer doesn't have a reputation for being extremely fast. But at least I turned off the TCP delay (slow-start algorithm), which is unsuitable for short connections, and the sending of the file is buffered. But even more important, all threads operate independently of each other. The native, and

therefore fast, accept() method decides which thread accepts a new connection. Apart from the ServerSocket object, the threads don't share any resources that might need to be synchronized. This solution is fast but, unfortunately, not very scalable, as threads are definitely a limited resource.

## Nonblocking Httpd

Let's look at another solution that uses the new I/O package. It's a bit more complicated and

requires the cooperation of different threads. It consists of four classes (see Figure 1):

1. NIOHttpd (see Listing 2)
2. Acceptor (see Listing 3)
3. Connection (see Listing 4)
4. ConnectionSelector (see Listing 5)

NIOHttpd basically launches the server. Just as in Httpd, a server socket is bound to port 8080. The important difference is that this time I use a java.nio.channels.ServerSocketChannel instead of a ServerSocket. I need to open the channel with a factory method before binding it explicitly to the port using the bind() method. Then I instantiate a ConnectionSelector and an Acceptor. Doing so, each ConnectionSelector is registered with an Acceptor. In addition, the Acceptor is provided with the ServerSocketChannel.

```

public static void main() throws IOException {
    ServerSocketChannel ssc =
    ServerSocketChannel.open();
    ssc.socket().bind(new
    InetSocketAddress(8080));
    ConnectionSelector cs = new
    ConnectionSelector();
    new Acceptor(ssc, cs);
}

```

Figure 2 depicts the concurrent execution of the Acceptor and ConnectionSelector threads. To understand the interaction between the two threads, let's first take a closer look at the Acceptor. Its task is to accept incoming connections and register them with the ConnectionSelector. Already in the constructor, the superclass's start() method is called as the required infinite loop is in the run() method. In this loop a blocking accept() method is called that will eventually return a socket object – almost exactly as in Httpd. But this time it's a ServerSocketChannel's accept() method, not a ServerSocket's. Finally, with the obtained socket-channel as an argument, a connection object is created and registered with the ConnectionSelector using its queue() method.

```

while (true) {
    ...
    socketChannel = serverSocketChannel.accept();
    connectionSelector.queue(new
    Connection(socketChannel));
    ...
}

```

To summarize: the Acceptor can only accept and register connections with a ConnectionSelector in an endless loop.

Like Acceptor, the ConnectionSelector is also a thread. In its constructor a queue is instantiated and a java.nio.channels.Selector is opened using the factory method Selector.open(). The Selector is probably the most important part of the server. It allows me to register connections and to obtain a list of those connections that are ready for reading or writing.

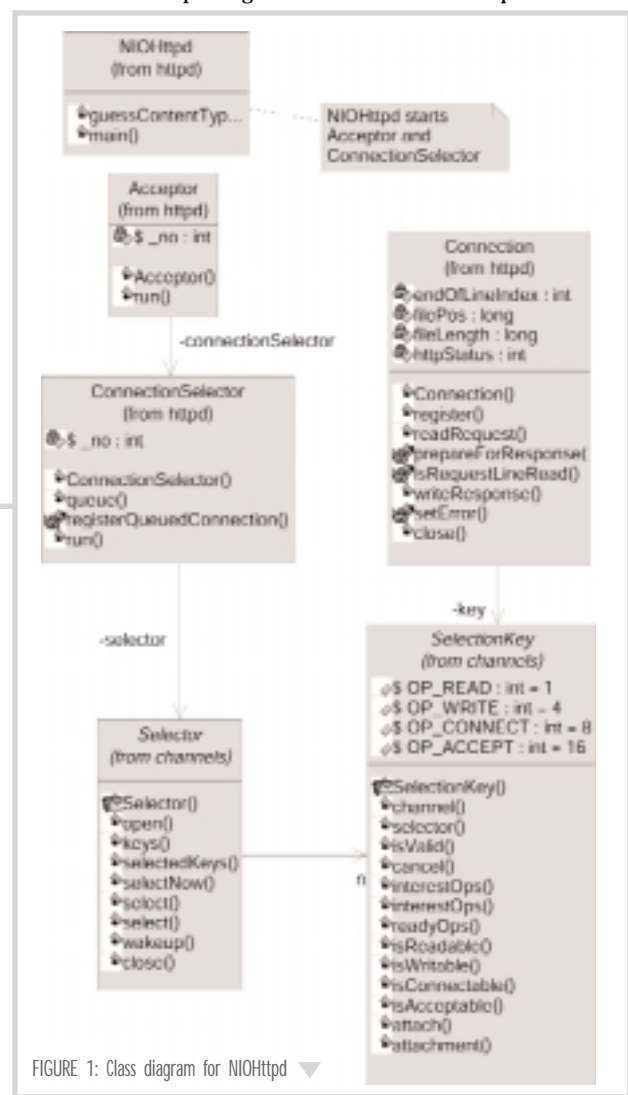


FIGURE 1: Class diagram for NIOHttpd

# Mongoose Technology

[www.portalstudio.com](http://www.portalstudio.com)



After the start() method is called in the constructor, the endless loop in run() is executed. In this loop I call the Selector's select() method. This method blocks until either one of the registered connections is ready for I/O operations or the Selector's wakeup() method is called.

```
while (true) {
    ...
    int i = selector.select();
    registerQueuedConnections();
    ...
    // handle connections...
}
```

It's crucial to understand that while the ConnectionSelector thread executes select(), no Acceptor thread can register connections with the Selector, because the corresponding methods are synchronized. Therefore I use a queue, to which the Acceptor thread adds connections as needed.

```
public void queue(Connection connection) {
    synchronized (queue) {
        queue.add(connection);
    }
    selector.wakeup();
}
```

Right after queuing a connection, the Acceptor calls the Selector's wakeup() method. This causes the ConnectionSelector thread to resume execution and return from the blocking select() call. Since the Selector is not blocked anymore, the ConnectionSelector can now register the connection from the queue. It happens the following way in registerQueuedConnections():

```
if (!queue.isEmpty()) {
    synchronized (queue) {
        while (!queue.isEmpty()) {
            Connection connection =
                (Connection)queue.remove(queue.size()-1);
            connection.register(selector);
        }
    }
}
```

### Selector Registration Using Keys

At this point I have to focus on the Connection's register() method. Until now I've talked about a connection that's registered with a Selector. This is a bit simplified. Instead, a java.nio.channels.SocketChannel object is registered with a Selector, but only for specific I/O operations. After registration, a java.nio.channels.SelectionKey is returned. This key can be associated with arbitrary objects using its attach() method. To get a connection with a key, I attach the Connection object to the key. By doing so I can indirectly obtain a Connection from the Selector.

```
public void register(Selector selector)
    throws IOException {
    key = socketChannel.register(selector,
        SelectionKey.OP_READ);
    key.attach(this);
}
```

Getting back to the ConnectionSelector, the select() method's return value indicates how many connections are ready for I/O operations. If the return value is zero, I skip the rest and return to the select() call. Otherwise, I iterate over the selection keys, which I obtained as Set by calling selectedKeys(). From the keys I get the previously attached Connection objects and call their readRequest() or writeResponse() methods. Which method is actually called depends on whether the connections were registered for read or write operations.

This eventually brings me back to the Connection class. It represents the connection and handles all the protocol's specifics. In its constructor the provided SocketChannel is set to nonblocking mode. This is essential for the server. Then a couple of default values are set and the buffer requestLineBuffer is allocated. As the allocation of direct buffers is somewhat expensive and I'm using a new buffer for each connection, I use java.nio.ByteBuffer.allocate() instead of ByteBuffer.allocateDirect(). If I reuse the buffer, a direct buffer could prove to be more efficient.

```
public Connection(SocketChannel socketChannel)
    throws IOException {
    this.socketChannel = socketChannel;
    ...
    socketChannel.configureBlocking(false);
    requestLineBuffer = ByteBuffer.allocate(512);
    ...
}
```

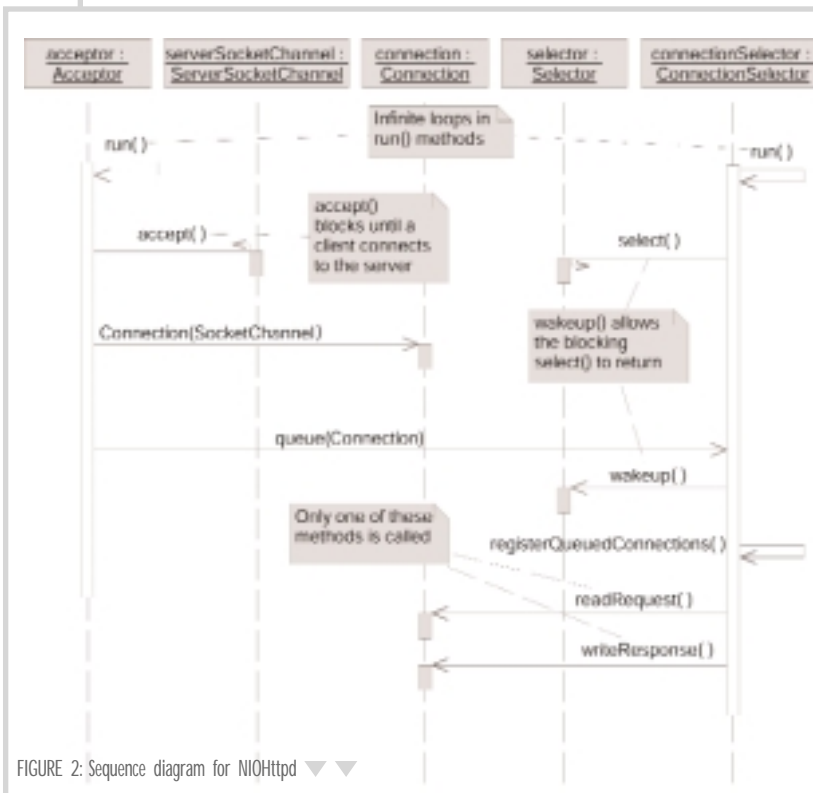


FIGURE 2: Sequence diagram for NIOHttpd

# Parasoft Corporation

[www.parasoft.com/jdj5](http://www.parasoft.com/jdj5)

After all initializations are done and the `SocketChannel` is ready for reading, the `readRequest()` method is called by the `ConnectionSelector`. Using `socketChannel.read(requestLineBuffer)`, all available bytes are read into the buffer. If the full line can't be read, I return to the calling `ConnectionSelector` and thus allow another connection to take over. However, if the whole line is read, it's time to interpret the request just as I did in `Httpd`. If it's a legitimate request, I create a `java.nio.channels.FileChannel` for the requested file and call the method `prepareForResponse()`.

```
private void prepareForResponse() throws
IOException {
    StringBuffer responseLine = new
StringBuffer(128);
    ...
    responseLineBuffer = ByteBuffer.wrap(
        responseLine.toString().getBytes("ASCII")
    );
    key.interestOps(SelectionKey.OP_WRITE);
    key.selector().wakeup();
}
```

`prepareForResponse()` builds the response line and (if necessary) headers as well as error messages, and writes this data to `responseLineBuffer`. This `ByteBuffer` is a thin wrapper around a byte array that was created using the factory method `ByteBuffer.wrap(byte[])`. After generating the data that I want to write, I need to notify the `ConnectionSelector` that from now on I want to write data rather than read it. This is achieved by calling the selection key's method `interestOps(SelectionKey.OP_WRITE)`. To guarantee that the selector quickly realizes the connection's change of interest, I call its `wakeup()` method.

Now the `ConnectionSelector` calls the connection's `writeResponse()` method. First, the `responseLineBuffer` is written to the socket channel. If the entire content of the buffer can be written, and if I still have to send the requested file, I call the `transferTo()` method of the `FileChannel` that I opened before. `transferTo()` potentially transfers data very efficiently from a file to a channel. How efficiently depends on the underlying operating system. In any case, only as many bytes are transferred as can be written to the target channel without blocking. To be on the safe side and to ensure fairness between connections, I set an upper limit of 64KB.

If all data is transferred, `close()` does the clean-up work. Here, the deregistering of the `Connection` is important. This is achieved by calling the selection key's `cancel()` method.

```
public void close() {
    ...
    if (key != null) key.cancel();
    ...
}
```

Again I wonder: How does this realization perform? And again I can answer: it performs well.

In principle, one `Acceptor` and one `ConnectionSelector` are sufficient to keep an arbitrary number of connections open. Thus this realization shines in the category of scalability. But as the two threads have to communicate through the `synchronized queue()` method, they might block each other. There are two ways out of this dilemma:

1. A better realization of the queue
2. Multiple `Acceptor/ConnectionSelector` pairs

One solution could be realized by using a `LinkedList` (see *Concurrent Programming in Java* by Doug Lea). This data structure is synchronized with two independent locks – one for the head and one for the tail. This ensures that adding and removing threads don't block each other. Only if the queue is empty is there a possibility of mutual blocking, but this can be avoided with an extra check.

In comparison to this elegant approach, my second solution qualifies for the "brute force" category. The load is balanced with multiple `Acceptor/ConnectionSelector` pairs and the synchronization problem isn't solved, but is somewhat reduced. Unfortunately, this causes additional costs for context switches. Compared to `Httpd`, fewer threads are needed.

One disadvantage to `NIOHttpd`, in comparison to `Httpd`, is that for each request, a new `Connection` object with buffers is created. This leads to an additional CPU cycle burning caused by the garbage collector. How large these extra costs are depends on the VM. However, Sun doesn't tire of emphasizing that with `Hotspot`, short-lived objects are not a problem anymore.

### Comparative Number Games

How much better does `NIOHttpd` scale than `Httpd`? Let's play with a couple of numbers, but before I go into media res, be warned: the formulas and the numbers I'm going to find are highly speculative. Only the concepts' performance is estimated. Important context variables like thread synchronization, context switches, paging, hard disk speed, and caches are not considered.

First I estimate how long it takes to process  $r$  simultaneous requests for files with size  $s$  bytes, if the client bandwidth is  $b$  bytes/second. In the case of `Httpd`, this obviously depends directly on the number of threads  $t$ , as only  $t$  requests can be processed at a time. I assume that a corresponding formula looks like Formula 1.  $c$  is the constant cost for parsing, etc., that has to be paid for every



B IN BYTES/S	S IN BYTES	D FOR $r \rightarrow \infty$	1/D
8000	1000	0.89	1.13
8000	10000	0.44	2.25
8000	100000	0.07	13.5
8000	1000000	0.01	126
10000000	1000	1.00	1.00
10000000	10000	1.00	1.0
10000000	100000	0.99	1.01
10000000	1000000	0.91	1.1

TABLE 1: d for  $r \rightarrow \infty$  with  $c=10ms$  and  $t=100$  ▼

# Interland

[www.interland.com](http://www.interland.com)

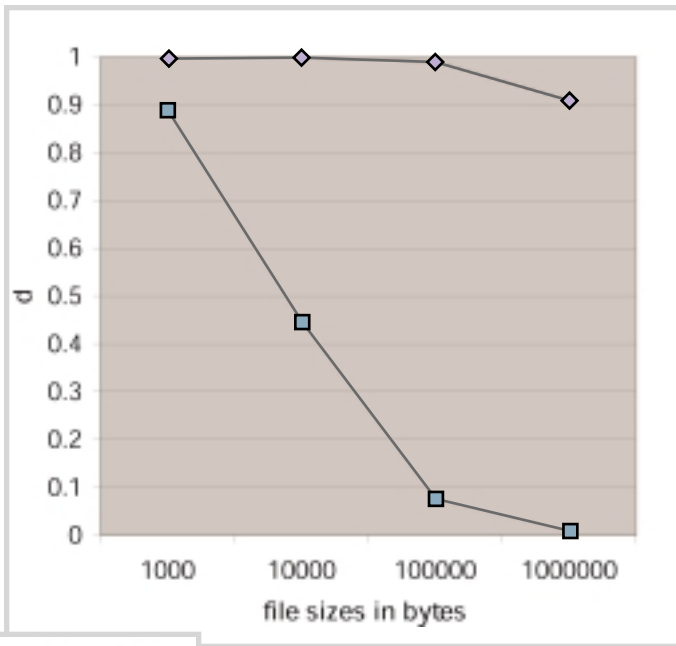


FIGURE 3:  $d$  for  $r \rightarrow \infty$  with  $c=10\text{ms}$  and  $t=100$

request. In addition, I assume I can read data faster from the disk than I can write it to the socket, my bandwidth is greater than the sum of the clients' bandwidth, and the CPU is not fully utilized. Therefore the server-side bandwidth, caches, and hard disk speed are not part of the equation.

FORMULA 1

$$l_{\text{Httpd}} = \frac{s \times r}{b \times \min(t, r)} + r \times c$$

However, NIOHttpd is not dependent on  $t$ . The transfer time  $l$  depends mostly on the client bandwidth  $b$ , the size of the file  $s$ , and the previously mentioned constant costs  $c$ . This leads to Formula 2, which estimates the minimum transfer time for NIOHttpd.

FORMULA 2

$$l_{\text{NIOHttpd}} = \frac{s}{b} + r \times c$$

The quotient  $d$  (see Formula 3) is of interest since it measures the relationship of the performances of NIOHttpd and Httpd.

FORMULA 3

$$d = \frac{l_{\text{NIOHttpd}}}{l_{\text{Httpd}}}$$

After closer examination (...and some rows of data), it becomes apparent that if  $s$ ,  $b$ ,  $t$ , and  $c$  are constant,  $d$  grows toward a limit. This limit can be easily calculated using Formula 4, which measures the limit of  $d$  for  $r \rightarrow \infty$ .

$$\lim_{r \rightarrow \infty} d_r = \frac{c}{c + \frac{s}{bt}} \quad \text{FORMULA 4}$$

Thus, besides the number of threads and constant costs, the connection's length  $s/b$  has tremendous influence on  $d$ . The longer the connection exists, the smaller  $d$  is, and the advantage of NIOHttpd compared to Httpd is greater. Table 1 and Figure 3 show that NIOHttpd can be 126 times faster than Httpd, given that  $c=10\text{ms}$ ,  $t=100$ ,  $s=1\text{MB}$ , and  $b=8\text{KB/s}$ . NIOHttpd has a big advantage if the connection stays open for a long time. If the connection is short, e.g., in a local 100Mb network, the advantage is only 10% provided the files are large. If the files are small, the difference won't be detectable.

In these calculations it's assumed that the constant costs of NIOHttpd and Httpd are about the same and no new costs are introduced by the different ways the servers have been implemented. As mentioned before, this comparison only holds under ideal conditions.

This is sufficient, however, to give you the idea that either concept might be beneficial. It should be noted that most Web files are small, but HTTP-1.1-clients try to keep the connection open as long as possible (with a keep-alive or persistent connection). Often, connections that will never again transfer any data are kept open. In a server with one thread per connection this leads to an incredible waste of resources. So, especially for HTTP servers, the scalability can be increased dramatically by using the new I/O API.

### Conclusion

With the new I/O API you can build highly scalable servers. In comparison to the old API, it's a bit more complex and requires a better understanding of multithreading and synchronization. Also, the documentation needs improvement. But if you've gotten over these hurdles, the new API proves to be a useful and necessary improvement of the Java 2 platform. ☛

### References

- HTTP 1.1: [www.w3.org/Protocols/rfc2616/rfc2616.html](http://www.w3.org/Protocols/rfc2616/rfc2616.html)
- Lea, D. (1999). *Concurrent Programming in Java: Design Principles and Patterns. Second Edition.* Addison-Wesley. <http://gee.cs.oswego.edu/dl/cpj>

### AUTHOR BIO

Hendrik Schreiber works as senior consultant for innoQ Deutschland GmbH ([www.innoq.com](http://www.innoq.com)) in Ratingen, Germany. His main area of interest is the architecture of modern Java-based solutions using J2EE. He is a coauthor of *Java Server and Servlets: Building Portable Web Applications*, published by Addison-Wesley.

[hendrik.schreiber@innoq.com](mailto:hendrik.schreiber@innoq.com)





# Actuate Corporation

[www.actuate.com/info/jbjad.asp](http://www.actuate.com/info/jbjad.asp)

# JDBC 3.0 – Something for Everyone

## Important new features



WRITTEN BY  
JOHN GOODSON

**T**here was no ticker tape parade to accompany the release of the JDBC 3.0 specification, but many will be pleasantly surprised at its list of enhancements that include everything from performance-tuning options to support for extended-level database features.

This article describes, in detail, the new features that are available in JDBC 3.0 and explains why they are important.

The JDBC 3.0 specification was shipped as part of the J2SE 1.4 release early in 2002. The key goals of the JDBC Expert Panel were to align with the most important features of SQL99, combine all the previous JDBC specifications into a single document, provide a standard way to take advantage of native DBMS functionality, and improve scalability. The JDBC 3.0 specification contains a collection of useful new features, none of which, if taken individually, would be considered “major.”

### Transactional Savepoints

One of the more useful new features of JDBC 3.0 is transactional savepoints. Traditionally, database transactions have been “all or nothing” types of events. An application would start a transaction, insert some rows, do some updates, and either make the changes permanent (commit) or discard them all (roll back) – all the changes would be made permanent or none of them would be.

With JDBC 3.0, the transactional model is more flexible. An application might start a transaction, insert several rows, and then create a savepoint, which serves as a bookmark. Then, the application might continue by performing some if/then/else type of logic, such as updating a group of rows. With JDBC 2.0, the application at this point would have been forced to either commit or roll back all the changes. In contrast, a JDBC 3.0 application might contain logic that determines that the updates were a bad idea, but the initial inserts were okay. In this case, the application can roll back to the savepoint (the bookmark) and com-

mit the group of inserts as if the updates had never been attempted (see the Listing 1).

### Pooling Enhancements

Connection pooling existed in the JDBC 2.0 specification, but JDBC 3.0 provides a much finer granularity of control over the connection objects in the pool. The single most expensive operation in a database application is establishing a connection. With some databases, establishing a database connection can take up to nine network round-trips between the JDBC driver and the database.

Connection pooling essentially involves keeping a cache of database connection objects open and making them available for immediate use by any application that requests a connection. Instead of performing expensive network round-trips to the database server, a connection attempt results in the reassignment of a connection from the local cache to the application. When the application disconnects, the physical tie to the database server isn't severed; instead the connection is placed back into the cache for immediate reuse.

With JDBC 2.0 connection pooling, there were no tuning options. You either used connection pooling or you didn't. With JDBC 3.0, there's a finer level of granularity available to control the characteristics of the connection pool. For example, you can fine-tune the following options to allow for maximum performance and scalability:

- The minimum number of connections to keep in the pool
- The maximum number of connections to have in the pool
- The initial pool size
- How long connections can remain

idle before they're discarded from the pool

- How often the connection pool should be evaluated to see if it meets the configuration criteria

In addition to connection pooling tuning options, JDBC 3.0 also specifies semantics for providing a statement pool. Similar to connection pooling, a statement pool caches PreparedStatement objects so they can be reused from the cache without application intervention. For example, an application might create a PreparedStatement object similar to the following SQL statement:

```
select name, address, dept, salary
from personnel where empid = ? or
name like ? or address = ?
```

When the PreparedStatement object is created, the SQL query is validated for syntax and a query optimization plan is produced. The process of creating a PreparedStatement is extremely expensive in terms of performance, especially with some database systems such as DB2. Once the PreparedStatement is closed, a JDBC 3.0-compliant driver places the PreparedStatement in a local cache instead of discarding it. If the application subsequently attempts to create a PreparedStatement with the same SQL query, the driver can retrieve the associated statement from the local cache instead of performing a network round-trip to the server and expensive database validation.

One advantage of connection pooling tuning properties and statement pooling is that their benefits can be realized in existing applications without any code changes. Upgrading an environ-

# **InstallShield Software Corp.**

[www.installshield.com](http://www.installshield.com)

# One of the more useful new features of JDBC 3.0 is transactional savepoints

ment to include a JDBC 3.0-compliant application server or driver can improve the performance and scalability of deployed systems, because the JDBC pooling enhancements are not directly invoked by the application's components. Instead, the pooling features are transparently used by the JDBC infrastructure.

## Retrieval of Autogenerated Keys

Many databases have hidden columns (called *pseudo-columns*) that represent a unique key over every row in a table. For example, Oracle and Informix have ROWID pseudo-columns and Microsoft SQL Server provides identity columns. Using these types of columns in a query typically provides the fastest way to access a row, because the pseudo-columns usually represent the physical disk address of the data. Before JDBC 3.0, an application could only retrieve the value of the pseudo-columns by executing a Select statement immediately after inserting the data. For example:

```
Int rowcount = stmt.executeUpdate (
    "insert into LocalGeniusList (name)
    values ('Karen')"); // insert row

// now get the disk address - rowid -
// for the newly inserted row
ResultSet rs = stmt.executeQuery (
    "select rowid from LocalGeniusList
    where name = 'Karen'");
```

This method of retrieving pseudo-columns has two major flaws. The first is that retrieving the pseudo-column requires a separate query to be sent over the network and executed on the server. The second is, because there might not be a primary key over the table, the search condition of the query may not be able to uniquely identify the row. In this case, multiple pseudo-column values could be returned and the application may not be able to figure out which one was actually the value for the most recently inserted row.

An optional feature of the JDBC 3.0 specification is the ability to retrieve autogenerated key information for a row when the row is inserted into a table. This new functionality removes the drawbacks in existing implementations that we discussed, because a separate query is not required to retrieve the key and the application is not responsible for the logic to retrieve the key. For example:

```
Int rowcount = stmt.executeUpdate (
    "insert into LocalGeniusList (name)
    values ('Karen'),
    Statement.RETURN_GENERATED_KEYS);
// insert row AND return key

ResultSet rs = stmt.getGeneratedKeys
(); // key is automatically
available
```

The application contains a value that can be used in a search condition to provide the fastest access to the row and a value that uniquely identifies the row, even when no primary key exists on the table.

Some databases, like DB2, do not provide an autogenerated key such as a pseudo-column. Fortunately, the JDBC 3.0 specification also allows the application to ask for the columns it knows represent a key on the table. For example:

```
// insert the row and specify that
// the employee ID be returned as the
// key
Int rowcount = stmt.executeUpdate (
    "insert into LocalGeniusList (name)
    values ('Karen'),
    'employeeID');

ResultSet rs = stmt.getGeneratedKeys
(); // Karen's employeeID value
is now available
```

The ability to retrieve autogenerated keys provides flexibility to the JDBC developer and a way to realize performance boosts when accessing data.

## Updating BLOB and CLOB Data Types

The SQL99 specification introduced two new advanced built-in data types, BLOB and CLOB, that provide flexible and fast access to long binary and character data items, respectively. While JDBC 2.0 provided mechanisms to read BLOB and CLOB data, it lacked a standard update method for those types, resulting in several problems for JDBC application developers.

The first problem is that some JDBC driver vendors introduced proprietary update mechanisms for BLOB and CLOB data types. Using proprietary methods inside an API standard makes those applications nonstandard by definition. The second problem is that many JDBC application developers assumed that they could use existing JDBC methods, such as `setString`, to update the values. Again, this resulted in different behavior on a driver-by-driver case, ultimately making those applications nonstandard.

JDBC 3.0 introduces a standard mechanism for updating BLOB and CLOB data. Note that the BLOB and CLOB interfaces apply only to database systems, such as Oracle, DB2, and Informix, that support the new SQL99 BLOB and CLOB data types. Microsoft SQL Server and Sybase support long varbinary and long varchar data types (image and text) that are similar to the new SQL99 types, but they don't support BLOB and CLOB types. Many JDBC application developers mistakenly think that the BLOB and CLOB interfaces should be used for all types of long data; however, they should be used only with BLOB and CLOB data. The JDBC specification provides methods such as `setString` to deal with long varchar and long varbinary data. Listing 2 shows how to update the contents of a CLOB column.

## Multiple Open Result Set Objects

Data architects routinely build their business integrity rules into database system stored procedures. For example, instead of trying to remember all the tables that must be updated when a new employee is hired, developers commonly use a central stored procedure, perhaps one named `AddNewEmployee`, that can be called when a new employee is hired. Calling stored procedures can result in table updates and, also, queries being executed that return result sets.

Invoking a single stored procedure can result in multiple result sets being returned to the calling application. For example, a stored procedure named

# **LOOX Software Inc**

[www.loox.com](http://www.loox.com)

employeeInfo might return the following result sets with:

- The employee's address and personal information
- A list of the employee's staff if he or she is a supervisor
- A list of the projects on which the employee is working

With JDBC 2.0, an application calling the employeeInfo stored procedure would have to serially process the three independent result sets that were generated at execution. If the application needed to first report on the projects list – the third of the sequenced result sets – then it would have to retrieve and discard the first two result sets. This processing methodology is somewhat inflexible and involves programming overhead and complexity. It also doesn't allow simultaneous access to any of the result sets generated from calling a procedure. That restriction limits the flexibility of a JDBC developer.

JDBC 3.0 gives developers the flexibility to decide if they want concurrent access to result sets generated from procedures or if they want the resources to be closed when a new result set is retrieved (JDBC 2.0 compliant behavior). Listing 3 is an example that simultaneously processes results from the employeeInfo procedure.

### Miscellaneous Features

We've detailed several new JDBC 3.0 features that range from performance enhancements to development flexibility. There are other JDBC 3.0 enhancements, including the ability to retrieve parameter metadata (allows development flexibility), allowing stored procedure parameters to be called by name (simplifies programming), the ability to specify holdable cursors (improves performance), and more SQL99 advanced data-type alignment.

There's something in JDBC 3.0 that will make every JDBC developer happier and it's available now. Remember, JDBC has both required and optional features, and I've described some of each in this article. Before assuming that a feature is supported in your favorite JDBC 3.0-compliant driver, check the database metadata to make sure optional features are supported. JDBC 3.0 is an approved specification – it's time to utilize the new features. Good luck! ☘

[john.goodson@datadirect-technologies.com](mailto:john.goodson@datadirect-technologies.com)

#### Listing 1

```
Statement stmt = conn.createStatement ();

Int rowcount = stmt.executeUpdate ("insert into etable (event) values
('TMM')");

Int rowcount = stmt.executeUpdate ("insert into costs (cost) values
(45.0)");

Savepoint sv1 = conn.setSavePoint ("svpoint1");
// create savepoint for inserts

Int rowcount = stmt.executeUpdate ("delete from employees");

Conn.rollback (sv1);
// discard the delete statement but keep the inserts

Conn.commit;
// inserts are now permanent
```

#### Listing 2

```
// Retrieve the case history for incident 71164
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery (
    "select CaseHistory from Cases where IncidentID=71164");

rs.next(); // position to the row
Blob data = rs.getClob (1);
// populate our Blob object with the data
Rs.close();

// now let's insert this history into another table
stmt.setClob (1, data);
// data is Clob object we retrieved from the history table

int InsertCount = stmt.executeUpdate (
    "insert into EscalatedIncidents (IncidentID, CaseHistory, Owner)"
    + " Values (71164, ?, 'Goodson') ");

// we're done ... CLOB data is now in the database
```

#### Listing 3

```
// get ready to call the employeeInfo procedure
CallableStatement cstmt = conn.prepareCall ("{call employeeInfo (?)}");

Cstmt.setInt (1,71164);
// bind parameter info for employee with id 71164

Boolean RetCode = Cstmt.execute ();
// call the procedure
// For simplicity we'll bypass logic for the procedure possibly returning
update counts

// first result set will be discarded ... materialize it and immediately
move to the second
ResultSet DiscardRS = cstmt.getResultSet();
// materialize first result set

ResultSet EmpListRS = cstmt.getMoreResults ();
// by default, close DiscardRS
// the 2nd result set: list of employees that report to 71164 is now
available

ResultSet ProjectsRS = cstmt.getMoreResults (KEEP_CURRENT_RESULT);
// the 3rd result set is now materialized and we can simultaneously oper-
ate on both
// the employee list and the project list
```

Download the Code!  
www.java-developers-journal.com

#### AUTHOR BIO

John Goodson is the vice president of research and development for DataDirect Technologies. For nearly 10 years, John has worked closely with Sun and Microsoft on the development of database access standards, and is an active member of the Expert Panel for the JDBC specification evolution. John earned his BS in computer science from Virginia Tech.

# Quintessence Systems Limited

[www.in2j.com](http://www.in2j.com)





## Reflections

JASON R. BRIGGS J2ME EDITOR

JavaOne is over, and it's time to sit back and reflect...and to sift through the hundreds of press releases and announcements that ricochet around the Internet like balls around a pinball machine. While I couldn't be there myself, when I checked my e-mails each day, I felt as if I was there in spirit at least.

For me, the most significant news to come out of JavaOne was talk of Monty – Sun's next-generation virtual machine for mobile devices. Monty is touted as being up to 10 times as fast as the current KVM, so it has great potential for multimedia applications and the like when it finally appears on a shipping device.

Among a number of announcements from Motorola: they unveiled the i95c1, a clam-shaped, Java-enabled mobile phone with a 120x160 pixel, 256-color screen; and their Semiconductor Products Sector (SPS) announced support for Bluetooth and for ARM's Jazelle acceleration technology in their Embedded Reference Implementation for J2ME.

Metrowerks and AGEA announced the CodeWarrior Wireless Enterprise Studio – in their words, “the first wireless application development solution”; while Kada Systems and Metrowerks (again) partnered with an integration plan for their respective technologies.

Kada Systems also wins my award for the largest text-based press release to arrive in my inbox: 127K...well, mime-encoded HTML probably accounted for most it, but they still win the award. Kada introduced frameworks to optimize deployment of J2ME apps, announced that they had completed the certification process for both CLDC and MIDP, talked about yet another partnership – this time with Softwired (iBus/Mobile JMS) – and were demonstrating the combined technology at JavaOne. Kada also entered into a strategic alliance with Espial, which I believe means that Espial's browser and applications will be running on Kada's platform.

Unfortunately, I've been unable to find a “Market Speak”-to-English translator at Babelfish, so my apologies if I've translated incorrectly.

From Wedgetail Communications came news of a licensing deal with Sony for Wedgetail's JCSI Micro Edition SSL security software (designed for J2ME) to be implemented in Sony's new interactive digital TVs. Wedgetail is a Brisbane-based company, which is a bit of problem for me, since I was formulating a theory that most software development is done by people in northern European countries where they have winter for most of the year, and so have nothing better to do with their time. I'm waiting for confirmation from Wedgetail that they've outsourced most of their programming effort to a development studio in Antarctica. I mean, they're Australians! Shouldn't they be standing in front of the barbeque with a couple of bottles of Victoria Bitter or something?

...

One of the joys of returning to the country of your birth after a long hiatus is the job hunt. Never really an enjoyable experience, as expected, after the dot bomb it's worse than ever. I'm not talking about cooling my heels in office receptions while waiting to be interviewed, nor am I whining about the fact that it's difficult (but not impossible) to find an agent who actually knows what J2EE means (and in this country, nigh impossible to find one who has heard of J2ME).

No, I'm talking about car parking. I don't begrudge someone wanting to make a respectable profit, but here in Auckland one particular car park conglomerate charges such high fees you need to take out a small mortgage to park your car in their buildings. So it's necessary to hunt out those precious council car parks – where a few dollars rents the space.

Which brings me to my killer app for the stressed car park hunter. A peer-to-peer (perhaps) application linking all car park meters in the city (Java-enabled parking meters, of course) with a GPS/MIDP application on my

### Reflections

JavaOne is now well and truly over, and it's time to sit back and reflect...

by Jason R. Briggs

62

### Edge to Edge

The JXTA for J2ME implementation interoperates with other JXTA implementations and provides a simple API to help J2ME developers quickly write JXTA applications and services for small devices.

by Kuldip Singh Pabla

64

### Jini Surrogate as a Platform for J2ME Games

This article continues the surrogate architecture tour from Part 1 (*JDJ*, Vol. 7, issue 3) and introduces a method through which a J2ME device can use it.

By William Swaney

66

### OSGi: The Last Mile of Software Deployment

Meet the OSGi Service Platform, Release 2 specification.

by Peter Kriens

74

### Keep Mobile Data and Applications in Sync with Java

Java provides the foundation for always-available mobile applications.

by Jeff Capone

84

mobile phone that can tell me exactly where the nearest empty space is. Rather than carrying an annoying pocketful of coins, the mobile phone can credit the meter with the parking fee, and the meter can message the phone when the time limit is due to expire.

This would be expensive to implement, but just imagine the convenience. No longer would I drive for 15 minutes just to find a cheaper space. I'd go directly to the nearest available space (I could even be messaged when a space is about to become available). Better yet, the meters would not need to be emptied every few days (a savings in labor costs), and there are a number of ways car park “overstayers” can be handled that result in major cost savings – so the system could eventually pay for itself.

Better yet, those high-priced car park companies might have to rethink their prices... ☛

▼▼▼ [jasonbriggs@sys-con.com](mailto: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.”

# QUALCOMM Incorporated

<http://brew.qualcomm.com/ZJD4>

WRITTEN BY KULDIP SINGH PABLA

## Edge to Edge



Already, important technologies like the small-footprint J2ME are in place to help developers create applications for small devices. With J2ME, developers can exploit the established capabilities of Java to write applications that are secure, interoperate with existing solutions, and easily port to new devices, all without sacrificing the flexibility that Java gives us.

Almost in parallel, interest is also growing in using peer-to-peer (P2P) technologies for wireless. Made popular by services like Gnutella and Napster, P2P computing rejects the notion of centralized servers in favor of peer groups – users bound by a common interest in processing, sharing, and storing information. With P2P, we can create networks of systems with very high levels of redundancy, privacy, and performance. Until recently, however, the growth in P2P was threatened because there was

no standard platform upon which applications could be crafted.

Project JXTA, incubated at Sun Microsystems and then quickly placed into the community development process, has established an early lead in developing the infrastructure needed to streamline the development of P2P applications. Unlike other approaches, JXTA provides the building blocks for creating P2P services and applications. In addition, JXTA offers strong security and interoperability between applications through well-defined protocols.

How can JXTA technology be extended into the wireless space? Most wireless devices have severe limitations on available memory, processor performance, and power. Pagers and cell phones can't be expected to support a full-blown JXTA environment, at least not anytime soon.

Fortunately, we have experience dealing with these limitations. The Java developer community tackled this issue some time ago and came up with the Connected Limited Device Configuration (CLDC) and Mobile Information Device Profiles (MIDP), core class libraries and specialized APIs designed to work in the constrained environments of wireless devices. Despite their small size, CLDC and MIDP offer developers the tools they need to create powerful wireless applications that can interoperate with the most sophisticated Java solutions running on workstations, servers, and mainframes.

The JXTA developer community has taken a similar approach by defining a set of functions designed to work in concert with MIDP. We call it the JXTA for J2ME

With the next generation of wireless devices entering the market, the opportunities for Java developers are great. Analysts are predicting that the demand for wireless applications is set to explode, with over 170 million U.S. users subscribing to wireless services by 2005. Other countries are moving even faster. Fortunately, recent developments have left us well prepared to meet the inevitable demand for applications that this growth will create.

implementation. Once fully deployed, it will allow a MIDP device to participate in P2P activities with JXTA peers running on larger platforms. The community's goals are straightforward: (1) wireless P2P applications should be easy to use and easy to develop, (2) they should be small enough to be used with cell phones and PDAs, and (3) they should be interoperable with JXTA applications running on larger systems. A tall order, given the constraints of wireless systems.

The solution is simple. Rather than hosting a complete JXTA environment on a small wireless device, some of the work will be done elsewhere. JXTA programmers are already familiar with the notion of a JXTA relay, a peer designated to act on behalf of others. JXTA relays help route information across networks and through firewalls. We've extended this idea to create a JXTA peer service that can act on behalf of a wireless JXTA peer, store and forward messages when polled, and translate and condense JXTA XML advertisements (the lingua franca of JXTA) into a more compact form easily processed by wireless peers.

This approach is straightforward, yet it's a no-compromise way to allow J2ME developers to extend P2P into the realm of wireless. A prototype application was demonstrated at JavaOne Japan in November 2001, and work continues on a full implementation. For details on its status, visit <http://jxme.jxta.org>.

The JXTA for J2ME implementation interoperates with other JXTA implementations and provides a simple API to help J2ME developers quickly write JXTA applications and services for small devices. I see opportunities to create some exciting applications in areas like gaming, financial services, and instant messaging using this technology.

Check out the Project JXTA Web site at [www.jxta.org](http://www.jxta.org) to learn more about peer-to-peer computing, experiment with some sample applications, download the docs and code, and start riding the next big wave of applications development. Join us! ☪

# Northwoods Software Corporation

[www.nwoods.com/go/](http://www.nwoods.com/go/)

[kuldipsingh.pabla@sun.com](mailto:kuldipsingh.pabla@sun.com)

#### AUTHOR BIO

Kuldip Singh Pabla is the software engineering manager with the Project JXTA team at Sun, managing various projects including the JXTA for J2ME implementation. Before JXTA, he was the engineering manager for Java Embedded Server, J2ME division at Sun.

# Sprint PCS

<http://developer.sprintpcs.com>

# Jini Surrogate as a Platform for J2ME Games

## Using HTTP as a communications protocol

### Part 2



WRITTEN BY  
WILLIAM SWANEY

In Part 1 of this article (*JDJ*, Vol. 7, issue 3) I introduced the idea of using the surrogate architecture within Jini as a platform for J2ME games. I also showed how to start Madison, Sun's reference implementation, and how to connect to it with the provided device simulator.

This article continues the surrogate architecture tour and introduces a method through which a J2ME device can use it.

To briefly recap, the architecture allows any device, in our case a J2ME one, to connect to a Jini network through a surrogate object that represents the device in the network. The surrogate host provides a mechanism for the devices to register themselves and obtain a context that enables access to the underlying Jini infrastructure.

Another important reason for choosing the surrogate architecture is that it allows us to view the device as a single object from the network perspective. There's no single monolithic portal API or application running; instead, many different surrogate objects with unique capabilities and representing many potentially different devices are displayed in the Jini network.

#### How Should We Connect?

The surrogate architecture defines the term *connector* as a means by which a device will discover and register with a surrogate host. Possible connectors might include such protocols as Bluetooth, USB, or even traditional network ones. Madison comes with an IP interconnect, useful in many cases but it may not be the best one for our project. For several reasons I believe HTTP is the best protocol to use.

HTTP? It does have its disadvantages. Foremost of these – it's a re-

quest/response protocol. This means that either we settle for a one-way communication model where all communication with the device must come as a response to an initial request, or we have an HTTP server at both the device and the surrogate object. The second option is just not possible in J2ME, so we're stuck with the first.

The second disadvantage is that HTTP has no notion of "discovery," at least in the Jini sense. This means that our device must have knowledge, in the form of a URL, of where the surrogate host is beforehand. It would be nice to somehow find a surrogate host but we can hardly multicast the Internet looking for one.

The J2ME specification only requires support for HTTP connections, though implementations of J2ME may support other types of communication. By choosing HTTP we guarantee that any J2ME device will be able to work with our surrogate host connector. There have been many debates as to whether J2ME is powerful enough, as well as fears that using it results in the loss of powerful native functionality. Cross-platform code has its own set of disadvantages, but in our case the portability is worth the loss in functionality.

To see another advantage, let's think about our use case. Our goal is to create a games platform on mobile devices. Typically, this means many users connecting with our system simultaneously. It's not unrealistic to think of multiplay-

er games with thousands of players concurrently connected. In this case, HTTP scales quite well – after all, it's a proven technology already handling huge numbers of simultaneous requests. Thus our system should be able to handle more users making requests every few minutes or so rather than a group of users who are constantly connected over sockets. It may even be cheaper for the end user by minimizing the amount of airtime consumed.

Finally, even though I'm advocating HTTP as the protocol to communicate with the surrogate host, it doesn't have to be the protocol used to communicate with the surrogate object once it's instantiated. I'll continue to use HTTP in our project, but the surrogate architecture doesn't require this and other protocols could be used depending on the device's J2ME implementation.

#### Defining Our Protocol

In defining a surrogate HTTP protocol I'll rely heavily on what has already been defined for the IP connector. That specification is already in the review stage of the Jini Decision Process (see [www.jini.org/standards/](http://www.jini.org/standards/)), and since HTTP is built on IP, there are some similarities. In addition, doing so is what I consider being a "good Jini citizen": cooperating and contributing to a community that gives a tremendous amount to its members. By building on the IP connector work, we respect the standards already defined, ensuring that Jini

# Sitraka

[www.sitraka.com/jprobe/jdj](http://www.sitraka.com/jprobe/jdj)



## “The J2ME specification only requires support for HTTP connections, though implementations of J2ME may support other types of communication”

services in general will be able to communicate with each other and interoperate without difficulty.

As stated earlier we won't need the discovery protocols. We will, however, need a registration protocol that allows a device to register with the surrogate host. Examining the IP interconnect registration request protocol we can see that it will suit our needs as well for HTTP. The registration request protocol is sent by the device to the surrogate host and has the following format:

- int protocol version
- short length of surrogate code URL
- byte[] surrogate code URL
- int length of initialization data
- byte[] initialization data
- int length of surrogate code
- byte[] surrogate code

We'll also need a response to this, which is something that the IP interconnect doesn't define. The main reason for defining a response is that if we use HTTP to continue communicating with the surrogate object, we must somehow let the device know how to accomplish this. Why? Once the surrogate object is instantiated, we're no longer communicating with the surrogate host. Instead, we're now communicating directly with the surrogate object. If HTTP is the only available protocol, then the device must initiate all communication. We define a response protocol in order to pass the URL for the surrogate object to the device, a different URL than for the surrogate host. Our response protocol is defined as follows:

- int protocol version
- short length of surrogate connect URL
- byte[] surrogate connect URL

Finally, since I intend to continue to use HTTP as the transport for device-to-surrogate object communication, it makes sense to define some sort of message protocol. Even though the type of messages sent may be very different depending on the type of game being played, I'll define a basic message proto-

col. An example of basic communication is passing a message that includes some data as a payload. Of course, the device and surrogate object must both be aware of the message format, though the surrogate host need not be. The message protocol might have the following format:

- int protocol version
- int message type
- int length of payload data
- byte[] payload data

### Finally Some Client Code . . .

Wait a minute, isn't this supposed to have something to do with J2ME? Well, there are still some missing pieces on the Jini side, but let's take a little break first and make use of the protocols introduced earlier.

Every game written using the HTTP interconnect will need to register with the surrogate host and receive a response. Therefore, it makes sense to encapsulate these two protocols in objects. Since our registration protocol is the same as the one for the IP interconnect, we may be able to reuse some of our code should we use J2ME over IP in the future.

The limited resources of a typical J2ME device now force us to make some design decisions. It's tempting to define an interface for our two protocols, something like the following:

```
public interface InterconnectProtocol
{
    void read(java.io.InputStream in)
    throws java.io.IOException;
    void write(java.io.OutputStream out)
    throws java.io.IOException;
}
```

This allows us a nice layer of abstraction, and by encapsulating the protocol in an object that accepts streams for reading and writing, we can use any type of connection.

What about those limited resources? By creating an interface we have one more file to add to our J2ME JAR. With some phones having an application

limit of 50K, why add unnecessary bloat? It's a case of evaluating each situation. For now I'll keep the interface, as it allows flexibility in adding more protocols for different message formats. Be aware though that sometimes every little byte counts; I've worked on applications in which I removed all evidence of object-oriented design in order to bring the size of the code down.

A further reduction in code size can be obtained, in our case by combining the registration and response protocols into one object. The reason is simply that the J2ME device need only send registrations to the surrogate host; it doesn't need to read the registration protocol. Similarly it only has to read the responses and never needs to send them. Again this is a decision to make based on the situation. By combining the protocols here we lose the reusability of the registration protocol with an IP interconnect, as the response is only needed with the HTTP interconnect.

See Listings 1 and 2 for J2ME encapsulations of the registration and response protocols, respectively.

### What Should an HTTP Surrogate Look Like?

Let's turn back to the Jini side of our project and think about what still needs to be defined, what an HTTP surrogate will look like, and how the surrogate host combined with the HTTP interconnect will interact with these surrogate objects.

In this section I'm assuming we'll continue to communicate with the surrogate object using HTTP. This means that either the surrogate must have an embedded HTTP server in its code, or the HTTP interconnect must provide that service for the surrogate object. The second case is not unreasonable as it simplifies the creation of surrogate objects, which is an important goal in a development platform. It also philosophically fits with the surrogate architecture: the host provides a context for the surrogate object and so does the interconnect.

This leads to the question of how a message is delivered from the interconnect-provided HTTP server to the surrogate. The embedded HTTP server provided by the interconnect and the surrogate object should have an exclusive relationship. At the very least, there should be some separation, and depending on the embedded server this might be an exclusive handler for HTTP requests destined for a specific surrogate object. To pass the message to the surrogate object we need to create an interface that it should implement:



# Motorola

[www.motorola.com/developers/wireless](http://www.motorola.com/developers/wireless)

## By creating an interface we have one more file to add to our J2ME JAR. With some phones having an application limit of 50K, why add unnecessary bloat?

```
public interface HttpSurrogate
extends net.jini.surrogate.Surrogate
{
    void handle(java.io.InputStream in,
java.io.OutputStream out) throws
java.io.IOException;
}
```

Are there potential problems in defining an interface our surrogates must implement? Other HTTP interconnect implementations may have different designs and we run into a case of surrogate objects being incompatible across different surrogate hosts. However, as we are going to further define the relationship between the HTTP surrogate object and the HTTP interconnect, this seems a small issue.

The `HttpSurrogate` interface extends the basic surrogate interface as defined in the surrogate architecture specification (included in the Madison download). This interface is defined as follows and allows the surrogate host to provide basic life-cycle services to the surrogate:

```
package net.jini.surrogate;
public interface Surrogate {
    void activate(HostContext
hostContext, Object context) throws
Exception;
    void deactivate();
}
```

The first object in the `activate()` method is the `HostContext`, provided by the surrogate host. The second object is the context, provided by the interconnect, and it's passed as an object; since the surrogate host calls this method on the surrogate, it doesn't need to know the nature of the context.

We'll need to define a context for the HTTP interconnect. The context defined in the IP interconnect fits our needs and keeps some consistency between the interconnects. The interface for the context is:

```
public interface
HttpInterconnectContext extends
```

```
net.jini.surrogate.KeepAliveManagement
{
    byte[] getInitializationData();
    java.net.InetAddress getAddress();
}
```

This provides a way to retrieve both the initialization data we may include in the registration protocol and the `InetAddress` of the device. We may want the `InetAddress` in the future, as it will allow the surrogate to create connections back to the device should the J2ME

implementation on that device support them.

You'll notice that the interface extends another surrogate specification interface. This allows the interconnect to assist in defining and enforcing what it means to have a live connection between a device and a surrogate.

### Tell Me What You Think

So far we've covered a lot of ground and further detailed some of the ideas introduced in Part 1. We're still not there, but we're getting closer to achieving the goal of developing a game platform for J2ME devices in a Jini network.

I'm interested in hearing your opinions. Are there pieces missing, whether technical or logistical, to this puzzle? How does the Jini/J2ME platform compare to others you've used for interactive, wireless device applications? Your thoughts and suggestions are welcome. ☺

▼▼ [wrswaney@netscape.net](mailto:wrswaney@netscape.net)

#### Listing 1

```
/**
 * An encapsulation of an http connector registration protocol.
 *
 * @author William Swaney
 * @version 1.0
 */
import java.io.*;

public class HttpInterconnectRegistration implements
InterconnectProtocol
{
    public static final int VERSION_NUMBER = 1;

    private short surrogateUrlLength=0;
    private byte[] surrogateUrl;

    private int initializationDataLength=0;
    private byte[] initializationData;

    private int lengthOfSurrogate=0;
    private byte[] surrogate;

    private java.io.DataInputStream din;

    public HttpInterconnectRegistration()
    {
    }

    public void read(InputStream in) throws IOException
    {
        throw new IOException(); //We should create our own excep-
            tion here
    }

    public void write(OutputStream out) throws IOException
    {
        if (out == null) throw new IOException();

        if ((surrogate == null && surrogateUrl == null) ||
(surrogateUrlLength == 0 && lengthOfSurrogate == 0))
            throw new IOException();

        DataOutputStream dataOut = new DataOutputStream(out);

        dataOut.writeInt(VERSION_NUMBER);

        dataOut.writeShort(surrogateUrlLength);
        if (surrogateUrlLength > 0) dataOut.write(surrogateUrl);

        dataOut.writeInt(initializationDataLength);
```

#### AUTHOR BIO

William Swaney, a software developer specializing in distributed computing, works for Valaran Corp., where he experiments with mobile devices and Jini networks.

# Ashnasoft Corporation

[www.ashnasoft.com](http://www.ashnasoft.com)

```

if (initializationDataLength > 0)
    dataOut.write(initializationData);

dataOut.writeInt(lengthOfSurrogate);
if (lengthOfSurrogate > 0) dataOut.write(surrogate);

dataOut.flush();
}

public void setSurrogateUrl(byte[] surrogateUrl)
{
    this.surrogateUrl = surrogateUrl;
    surrogateUrlLength = (short)this.surrogateUrl.length;
}

public void setSurrogateUrl(String surrogateUrl)
{
    setSurrogateUrl(surrogateUrl.getBytes());
}

public void setInitializationData(byte[] initializationData)
{
    this.initializationData = initializationData;
    initializationDataLength = this.initializationData.length;
}

public void setSurrogate(byte[] surrogate)
{
    this.surrogate = surrogate;
    lengthOfSurrogate = this.surrogate.length;
}
}

```

Listing 2

```

/**
 * An encapsulation of the response message sent from the http
 * connector to the device.
 *
 * @author William Swaney
 * @version 1.0
 */
import java.io.*;

public class HttpInterconnectResponse implements

```

```

InterconnectProtocol
{
    private int protocolVersion;
    private short surrogateConnectUrlLength=0;
    private byte[] surrogateConnectUrl;

    private java.io.DataInputStream din;

    public HttpInterconnectResponse()
    {
    }

    public void read(InputStream in) throws IOException
    {
        din = new DataInputStream(in);

        protocolVersion = din.readInt();

        surrogateConnectUrlLength = din.readShort();
        surrogateConnectUrl = new byte[surrogateConnectUrlLength];
        din.readFully(surrogateConnectUrl);
    }

    public void write(OutputStream out) throws IOException
    {
        throw new java.io.IOException(); //We should create our
        own exception here
    }

    public int getProtocolVersion()
    {
        return protocolVersion;
    }

    public short getSurrogateConnectUrlLength()
    {
        return surrogateConnectUrlLength;
    }

    public byte[] getSurrogateConnectUrl()
    {
        return surrogateConnectUrl;
    }
}

```

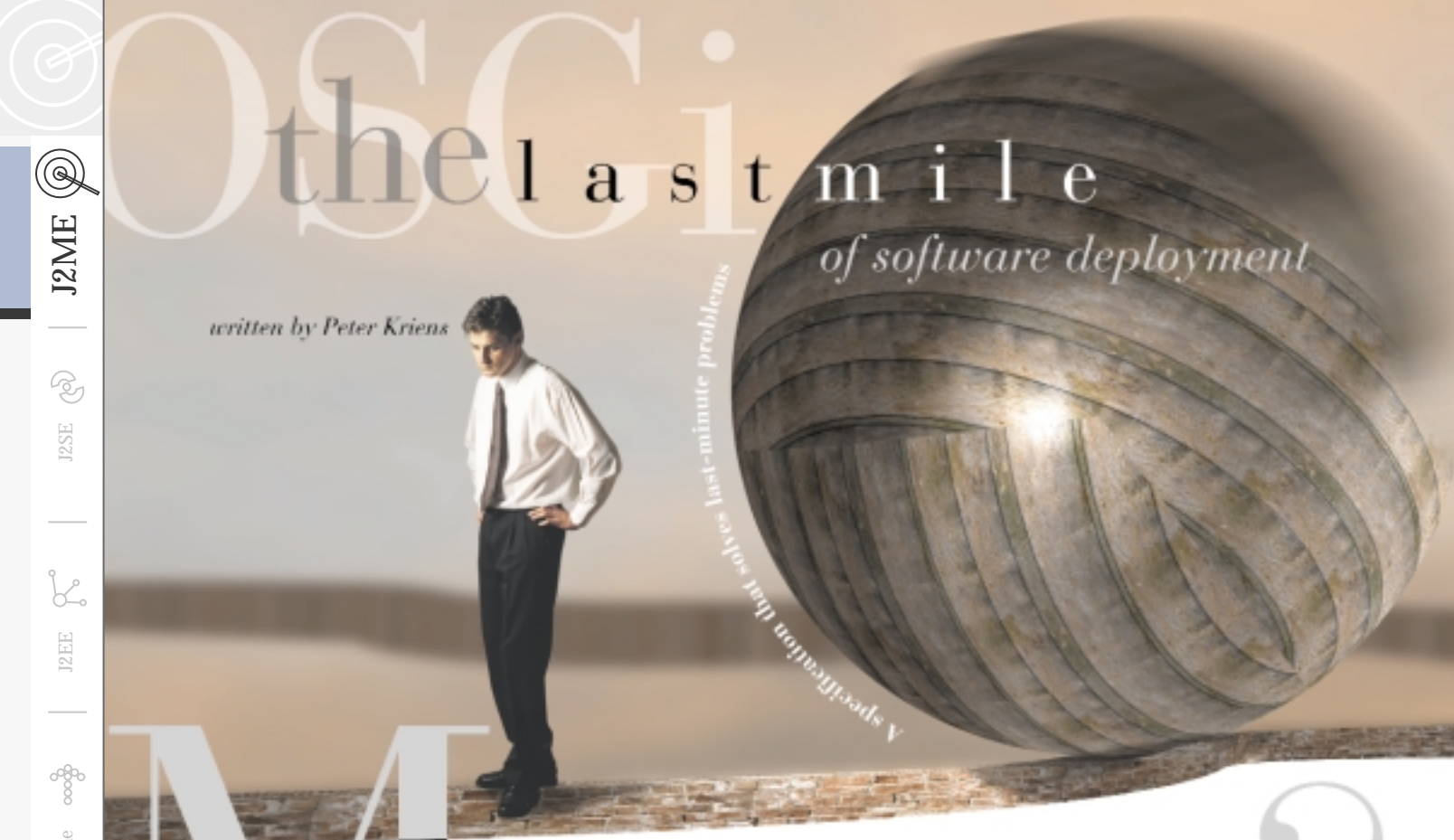


# AppDev Training Company

[www.appdev.com/promo/MG00052](http://www.appdev.com/promo/MG00052)

# New Atlanta Communications

[www.newatlanta.com](http://www.newatlanta.com)



# The Last Mile of software deployment

written by Peter Kriens

A specification that solves last-minute problems

# M

aking good software is hard enough, as most software developers can testify. There is, however, one aspect of software that is invariably even harder than we expect: How do we get the software into the target environment?

When the rubber finally meets the pavement, we usually find many complicated and awkward problems. Differences in hardware, operating system configurations, or setup information can bring our carefully designed and crafted software to a halt in an unexpected environment. Other common problems are versions that don't match, missing packages or libraries, different names for environment properties, and so on.

Most of us who have had to write installation programs or scripts are aware of the sometimes embarrassing amount of time needed to get an already-working software system operational on different target machines. I remember a colleague who promised to eat a floppy disk if the installation script failed one more time, and was consequently served a tossed floppy salad a few hours later...with dressing.

Meet the OSGi Service Platform, Release 2 specification. This specification is the result of two years of intensive collaboration between OSGi members (Sun, IBM, Ericsson, Nokia, Motorola, Oracle, Telcordia, HP, and 70 others). It has already been implemented by several vendors, and development versions can be downloaded from Gatespace, IBM, ProSyst, Sun, and others. It allows independent application developers to develop software that can be remotely managed for different standalone computers, processors, and operating systems of various sizes and configurations. This list includes, but is definitely not limited to, embedded computers.

The specification relies heavily on Java, the obvious choice when the final platform can't be restricted to Microsoft

Windows hardware requirements, or where high reliability and availability is needed. On top of Java, the specification defines a framework that addresses the needs of a platform that's running continuously, and must run a dynamically changing set of applications, and react directly - without supervision - to changes in its environment.

A key concept in the framework to achieving these desired properties is the *service*. A service is a Java object that's registered with the framework and is to be used by other applications. The functionality of a service is defined by the interfaces it implements. This allows many different applications to implement the same service type. For example, a service for logging could be optimized for local storage and another implementation could use a remote management system. For the user of this log service, there should be no detectable difference.

On top of the framework, the OSGi has defined a number of services that can be used by applications. This rapidly expanding list contains the following services:

- **Logservice**: Provides a way to log information from applications and a way to receive the log entries while they're being made.
- **HTTP service**: Provides a Web server (HTTP or HTTPS, depending on the implementation) that supports servlets and static files. Combined with the other features of the framework and services, this is probably one of the easiest ways to deploy Java servlets.
- **Device access**: Supports the automatic download of device drivers, allowing for true plug and play.



# Fiorano Software

[www.fiorano.com/tifosi/freedownload.htm](http://www.fiorano.com/tifosi/freedownload.htm)



- **Configuration management:** Allows management systems or applications to set the configuration parameters of applications in real time, without requiring a restart when the parameters have changed. The specification uses a metatyping specification that allows applications to discover the (localized) details of data types that are needed for a particular application.
- **Preferences:** Supports a hierarchy of properties. This simplified database can be stored locally or on a remote back end.
- **User admin:** Provides a repository of users and includes authentication and authorization of those users. Many different authentication systems are supported.

The following are some of the services that will probably be added in the next release:

- **Wiring admin:** Provides a comprehensive model to connect different services into larger function blocks. This allows simple scenarios, such as connecting a switch to a light, but can also be used to connect a GPS to a navigation system. Filters can be used to do conversions or thresholding.
- **Position service:** Initially targeted at the car industry, it was found to be useful in other contexts as well. Just think of an electronic vacuum cleaner in your residence.

This list will expand significantly in the near future. OSGi has many expert groups working in certain areas. These groups adopt existing standards or develop derived specifications that add to the list of available services. Other groups are working on adopting OSGi specifications or developing their own specifications for vertical markets. The OSGi is open and invites companies to participate in this specification process (see [www.osgi.org](http://www.osgi.org)).

Applications can also use standard Java libraries. These libraries can be wrapped in the application's delivery file or referred to from the manifest file. Versioning issues are explicitly managed by the framework.

Figure 1 shows how the different parts are related. It also shows that bundles (the OSGi applications) can use the OSGi Framework and all standard Java libraries, and access the operating system with the underlying hardware.

A key aspect of the OSGi is that there are numerous vendors (e.g., Gamespace, IBM, ProSyst, and Sun) that implement the specifications while they're being developed. In addition, the OSGi expert groups develop a reference implementation and test suites for each specification. This means that the specifications are practical, usable, and have multiple (choice!) industrial-strength implementations available (most

“A key aspect of the OSGi is that there are numerous vendors that implement the specifications while they’re being developed”

- **Message- and connection-based communication:** A special federated address concept that allows OSGi applications to communicate transparently (under full security control, obviously) even when they're separated by multiple firewalls or use different communication technologies (e.g., IP and SMS of mobile phones).
- **Certificate support:** Simplifies the validation of certificates and provides access to the certificates of the local or remote management systems.
- **Execution environment:** One or more definitions of what classes and methods minimal environments should support.

of the vendors actually have free downloadable SDKs for evaluation purposes). An open source implementation called *Oscar* (that still has some work to do, especially to match release 2) can be found at <http://oscar-osgi.sourceforge.net>.

### The Framework

The framework is the part that changes a Java Virtual Machine from a single application environment into a multiple one. The advantages are many: running multiple applications in a single VM means fewer process swaps, fast inter-application communication, and significantly less memory consumption. Protection between the applications is provided by the Java runtime and the framework.

The concept of a bundle is used to represent these multiple applications. A bundle is a Java ARchive (JAR) file containing “parts” that are needed to run the application, and a manifest file declaring the parts that should be available when the application is started. These parts are provided by other bundles or by the environment. Dependencies are fully managed by the framework. When bundles are started, they usually register and get services. Services are Java objects defined by a Java interface that can be found in the framework service registry. Bundles can communicate only through these services (see Figure 2).

### Management Bundle

An important concept of the OSGi specifications is the so-called “management bundle.” This is a bundle with administrative privileges. Management bundles are required because the framework is “policy free,” i.e., it provides many mechanisms but carefully tries to abstract itself from decision-making. These decisions, the policies, are provided by the man-

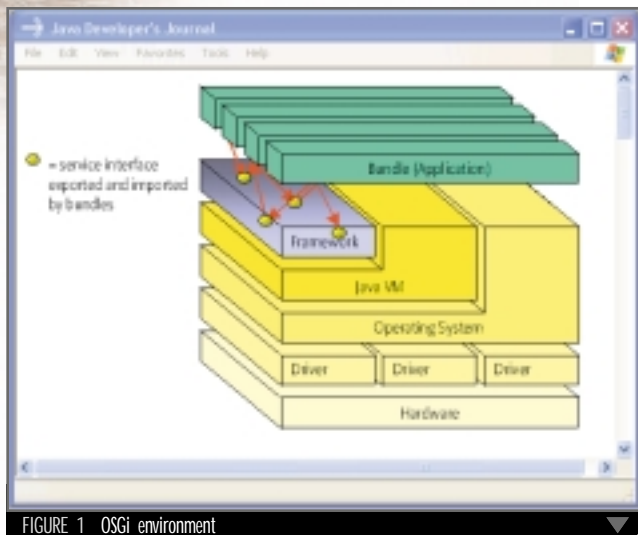


FIGURE 1 OSGi environment

# eXcelon Corporation

[www.exln.com](http://www.exln.com)

agement bundle and will vary from installation to installation. Management bundles are responsible for managing the environment. Many different types exist: command-line consoles, remotely managed SNMP-based bundles, Web-based interfaces, and static configuration-based management bundles.

One of the management bundle's many responsibilities is installing bundles. An API is provided that gives the framework a Java InputStream object and a name. The InputStream is connected to a JAR file that contains the classes and resources that are provided by that bundle. This fits seamlessly with the Java URL and file system model. Uninstalling doesn't require a complicated uninstall script – the framework is fully aware of all the aspects that are part of the bundle and can completely clean up after itself. It even cleans up the private files that were generated by the bundle. Updating the bundle is just as simple.

### Package Dependencies

Once a bundle is installed it needs to be started. However, there are dependencies specified in the JAR file that need to be resolved. These dependencies are about packages. Java classes are always contained in a package. This is the first part of the name of the class (until the last “.”), e.g., the package of `java.lang.String` is `java.lang`.

A bundle can use packages in three different ways. First, it can have private packages. All classes of the bundle that don't need to be shared with other bundles should be private. Different bundles can thus include the same packages.

Second, classes that need to be shared with other bundles should be exported. Interfaces and classes that are used to communicate (e.g., the interfaces that define the services) must be exported or else the bundles will run into `ClassCastException`s when they exchange objects. Any bundle can specify any package for export. The framework will ensure that only one of the bundles at any time will export a specific package.

Third, the bundle can also specify that it needs certain packages; for example, that the classes in its JAR file refer to the `javax.servlet` and `javax.servlet.http` packages. Imports require an associated export. The framework will pick only one bundle to export a specific package at any time to prevent `ClassCastException`s. The decision of which bundle to pick for export is further guided by the versions (all package references can contain a version specifier) and security.

Figure 3 shows how packages can be

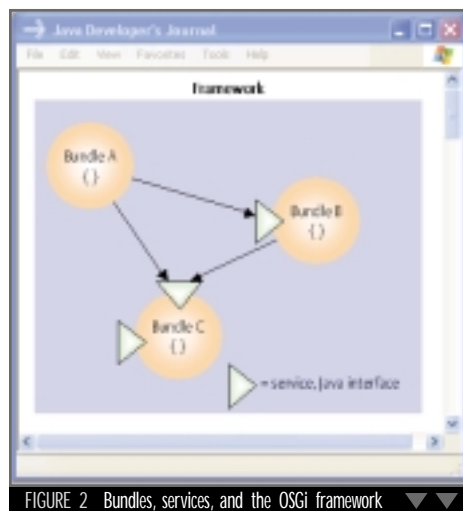


FIGURE 2 Bundles, services, and the OSGi framework

and `javax.servlet.http` packages. Managing the classpath is one of the framework's most important tasks. Some people have said that the OSGi specifications deliver the original promise of classloaders and classpaths.

### Native Libraries

The JAR files can also contain native libraries. These libraries are not restricted to a single hardware/OS architecture. The manifest can contain a specification of which libraries are intended for which architecture. The framework will find the set of possible libraries and load the best-fitting one. The framework will handle all complicated library path issues that normally plague the deployment of native code.

### Starting the Bundle

After the dependencies are resolved and the right bundles are picked to export, the bundle needs to be started. For this reason, the manifest contains a header that points to a special class: the activator. An object of this class is created and cast to a `BundleActivator`. This interface contains a `start` method and a `stop` method that are used to start and stop the bundle. For efficiency reasons, these methods should return quickly or they'll block the system. They normally start a background thread or `get/register` some services. The `stop` method cleans

“The framework is the part that changes a Java Virtual Machine from a single application environment into a multiple one”

imported from the framework (or standard Java classpath) or other bundles, or exported. The resolving of these packages is under the control of the framework and may be subject to security checks.

Bundles can also include other JAR files. The manifest can be used to tell the framework that this JAR file contains a library that must be available on the classpath. For example, a bundle could include the `servlet.jar` JAR file and then export the `javax.servlet`

up all the parts that the framework can't directly clean up and stops any threads.

### Service Registry

A started bundle is expected to provide some utility to the end user. It's highly likely that it's using services to provide this functionality to users and other bundles. The services – Java objects implementing a service interface – are available from the framework registry. This registry is searchable with a simple but very expressive filter. The syntax of this filter is derived from the LDAP filter syntax RFC-1960. This type of filter can have comparisons for equality (`=`), magnitude (`<=`, `>=`), substring

# Borland Software Corp.

[www.borland.com/new/optimizeit/94000.html](http://www.borland.com/new/optimizeit/94000.html)



(\*x\*), presence (\*), and approximate matches (~=). Expressions can be AND (&), OR (|), NOT (!), and nested to any depth.

For example:

```
( & (service.pid=USB-1232312452)( |  
(vendor~=ericsson)(vendor~=ibm) ) )
```

This filter language is used to search the framework, to pre-filter events before they're delivered, and, in many service specifications, to find objects.

The properties that are used in a search are given to the framework when a service is registered. Some properties (like the interfaces that the service implements) are automatically set by the framework, some properties have defined semantics (like "service.pid" is a unique identifier for a service), and other properties are defined by the bundles themselves. Each property can, as with LDAP, have multiple values; for example, a device needs to be registered with a property DEVICE\_CATEGORY. However, a device that fits multiple device categories can put the categories in an array or a vector object.

### Dynamism

A cornerstone of the OSGi architecture is dynamism. Bundles can be installed, updated, started, stopped, and uninstalled at any time; devices can come in any range and be represented in the framework registry at any time. This makes the environment extremely dynamic. Therefore, the framework offers a comprehensive model for writing applications for such an environment. All important changes to the environment are sent to listeners that have specified an interest in these events.

Installation, start, stop, and update of bundles is sent out as bundle events. Management bundles can use these events to apply their policies (for example, the security settings or installing required support bundles) at the appropriate time. Service registrations, unregistrations, and modification of service properties are sent as service events (see Figure 4).

Bundles use these events to adapt to the changes in the environment. For example, when a digital camera is plugged

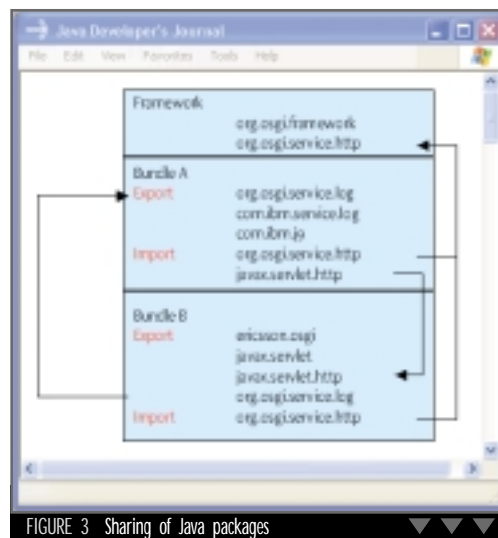


FIGURE 3 Sharing of Java packages

bundles, devices, and services available – a must for a stand-alone, continuously running server.

### Cleaning Up

Running a number of bundles together in a single VM and sharing the resources is not always easy; therefore, the framework closely tracks the dependencies between the bundles. When bundles are stopped or uninstalled, the framework will use its extensive knowledge of these dependencies to clean up as much as possible. Services registered by a bundle are automatically unregistered and services used from the framework are returned. This significantly minimizes the complexity of sharing a VM.

### Security

The framework has been targeted to run multiple independent bundles simultaneously. It was assumed from the beginning that these bundles couldn't all be trusted in such an environment; therefore, security is paramount. That's the rea-

“**A** cornerstone of the OSGi architecture is dynamism. Bundles can be installed, updated, started, stopped, and uninstalled at any time”

into a FireWire network, a device service representing this camera is registered. A bundle that controls a monitor can then react to this registration and add this new camera to the monitor's menu. A special class, the ServiceTracker, is provided to simplify this process of tracking services in the registry, which makes using this dynamic model trivial.

Though the dynamism undoubtedly introduces a certain amount of complexity into the programming, the benefits are huge. OSGi-based environments don't have to be rebooted when configuration changes take place or bundles are installed and updated. The environment adapts itself to the

son the OSGi has adopted the Java 2 security model. In this model, a certain "type" of security is represented by a Permissions class. For example, access to the file system is guarded by the FilePermission class. These permissions are associated with the code base, i.e., the place where the code came from. When a permission needs to be verified, the checker creates an instance of the appropriate Permission class and calls the SecurityManager with this object. This will assure that the stack is crawled, and each caller is checked to see if it has a Permission object. For example, when a file is opened, the Java IO runtime system creates a FilePermission object with the filename as a parameter and with READ as the action. The caller (and all its callers) are then checked to see if its set of permissions include a FilePermission object that implies the given filename and action.

Normally, Permission objects are associated with code bases through the Policy class. The default policy reads the

# Pramati Technologies

[www.pramati.com](http://www.pramati.com)



permissions from a file. Permissions can normally be granted on a file basis or when certificates are used on the signer basis. This is so flexible that it's sometimes quite confusing.

Therefore, the OSGi has simplified the model significantly by granting permissions only on a per-bundle basis. As is customary in the OSGi specification, the mechanism to associate the permissions with the bundle are made in such a way that an operator can use his or her own policies, bought from network management vendors or rolled by the operator's staff.

Management bundles have two mechanisms available for permission-handling. One, there's a special service, `PermissionAdmin`, that permits a management bundle to set, get, and reset the bundle and default permissions. The permissions are associated with the location (this is normally the URL where the bundle comes from). This makes it possible to set the permissions before the bundle is actually downloaded in the environment. However, there's also the possibility of just-in-time permission-setting.

Two, a management bundle can register a `SynchronousBundleListener` with the framework. This listener is called when the bundle is installed but before the bundle has had a chance to do something (including exporting code). At this moment, the management bundle can look up the permissions and set them.

As always, using secure systems doesn't make life easier for the developer, but at least the OSGi environment makes the

scale installations is complex and error-prone. Therefore, these tasks should be minimized as much as possible. Having one broad permission for administrative tasks was deemed more than sufficient and significantly reduces the administration of permissions.

Bundles can import and export Java packages as explained earlier. This is obviously a security risk because malicious bundles could export an innocent-looking package that's inadvertently used by an important bundle. Also the reverse, a bundle importing a package that it might use to do "evil" things should be under strict operator control. These threats are mitigated with the `PackagePermission` class. `PackagePermission` classes can hold the name of a package (including wildcards) and the action export or import.

As usual, some bundles are more equal than others and the `ServicePermission` class is intended to differentiate between them. Bundles can be given permission to register or get a specific service (or a wildcard). Conforming frameworks must ensure that when a bundle doesn't have this permission, it won't be able to detect the existence of such a service. For example, when a bundle registers an HTTP service object with the framework, another bundle that doesn't have `ServicePermission(HttpService,GET)` won't even be able to see the registration event of that service.

This is a powerful mechanism that's broadly used in the OSGi specifications to raise firewalls between bundles. For example, in the configuration management specification

# “**T**hough the dynamism undoubtedly introduces a certain amount of complexity into the programming, the benefits are huge

Java 2 security model significantly less painful to use in practice.

## Framework Permissions

The framework introduces three new permission types: `ServicePermission`, `PackagePermission`, and `AdminPermission`.

`AdminPermission` gives the owner the authority to manage the framework. Management bundles require `AdminPermission` to perform their numerous duties. This `AdminPermission` is a broad permission class intentionally. Administering and configuring large-

there are two parties: the Configuration Admin service that configures clients and the clients that are configured. Both are represented as services in the framework service registry. However, registering a Configuration Admin service is a privileged action, reserved for the bundle designated by the operator. On the other hand, registering as a client is allowed by basically every bundle. Getting the client service is again privileged and getting the Configuration Admin service is not.

It turns out that many security schemes don't require special Permission classes, but can leverage the expressive power of `ServicePermission`.

## Applications

The framework, though initially targeted at residential gateways, has already found a wider audience. The car industry is showing great interest in the OSGi specifications and is currently participating in continuing the effort. The applicability of the specifications is even wider than gateways in cars or homes. It can be used to deploy applications in almost any environment where they consist of multiple components that, together, form the needed functionality. The ease of remote management, a well-defined environment, numerous available tools, security, many industrial-strength SDKs, and the dynamic collaborative environment make it interesting for many software deployment problems, from PC-based applications to high-end Unix application servers, from servlet-based applications to cellular network base station controllers somewhere out in the countryside. Why not give it a try! ☛

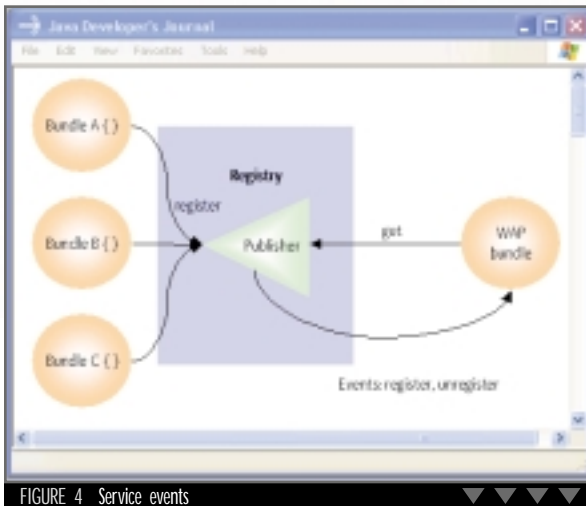


FIGURE 4 Service events

### AUTHOR BIO

Peter Kriens, an OSGi technical officer, also worked for Ericsson Research in Stockholm, where he got involved with home servers and residential gateways. Peter studied electronics in Alkmaar, Holland.

[peter.kriens@aquite.se](mailto:peter.kriens@aquite.se)

# InetSoft Technology Corp.

[www.inetsoft.com/jdj](http://www.inetsoft.com/jdj)

# Keep Mobile Data and Applications in Sync with Java

An integral part of your mobile strategy



WRITTEN BY  
JEFF CAPONE

**H**ere's a quiz: consider a physician accessing different patients' histories on a PDA while making rounds at a Manhattan hospital versus a field engineer whose responsibility is to monitor and repair sections of an oil pipeline that stretch across 200 miles in Texas.

Which professional is able to have continuous access to data and applications – the doctor, who is always connected to the wireless LAN network, or the field worker, who experiences only intermittent connectivity?

The answer is both. The doctor and the field engineer can have uninterrupted access to data and applications, without the unrealistic expectation of continuous wireless connectivity, if they're equipped with an "always-available" mobile solution.

Java provides the foundation for always-available mobile applications. For example, a J2ME-enabled mobile device can operate in both wireless and disconnected modes. Unlike browser-based WAP or HTML applications, J2ME applications are persistent; a J2ME application can function on a cellular phone, laptop, or PDA without a wireless connection. When a wireless connection is established, however, the J2ME application can communicate with the corporate network, sending and receiving data changes, additions, and deletions.

Java's ability to work in all modes makes this technology invaluable for all types of environments and mobile applications. In addition, the user interface of a J2ME application can be customized and enriched with graphics, media components, and more intuitive navigation to improve the user experience. The challenge, then, is to tailor a Java-based solution around the business needs, the nature of the work, and the mobile environment.

## Always-Available Mobile Applications

The need for mobile applications to

operate in any mode is the result of three overriding business concerns:

1. Limitations in network infrastructure
2. Low wireless connection bandwidth
3. Cost of wireless service

First, current limitations in network coverage make disconnected applications a must-have for the mobile professional. Whether a sales representative is traveling on a plane or a field engineer is in the desert fixing a pipeline, they must all be productive with or without the benefit of a wireless connection. Until wireless connections are ubiquitous and 100% reliable around the world, disconnected functionality will remain invaluable to the mobile worker.

Low wireless connection bandwidth is also an issue with most mobile professionals. It will continue to be a bottleneck, even with GPRS and 3G networks. Synchronizing applications only when necessary is therefore of great value. A field engineer, for example, may not want to update daily job information every time he or she enters one line of data; the engineer may simply need to ensure all jobs have been entered, verified, and completed intermittently throughout the day.

Airtime minutes will continue to be a cost factor until carriers and other wireless network providers switch to a true data usage-based model. Synchronization eliminates the needless overhead of "dead" airtime minutes used during a wireless data session in which a user is charged even though no data was exchanged during the wireless connection (such as when reading e-mail). The

beauty of synchronization is that it allows the user to connect to the network only if data transmission is necessary, and automatically disconnect when the transmission has been accomplished.

In this environment, choosing the appropriate synchronization model is critical to successfully deploying mobile applications.

## Mobility Achieved with a Java Synchronization Solution

There are several synchronization approaches. File sharing is one common option, but it requires the presence of databases on both the client and the server. This becomes cumbersome if all you want to do is synchronize data between enterprise and client applications, neither of which stores data in databases. It also often requires a proprietary file format for transporting the data between devices.

Application-level synchronization is another option, but its shortcomings include the inability to extend the synchronization solution across multiple applications and back-end databases. Since this synchronization solution is specific to the application, multiple solutions may need to be supported when a project requires access to more than one enterprise application.

An optimal synchronization solution would sit between the application and database level, allowing objects or components to freely synchronize with one another. This is possible using Java (see Figure 1). By allowing a server's data access layer to synchronize directly with the client's data-access layer, it doesn't

# Softwired, Inc.

[www.softwired-inc.com](http://www.softwired-inc.com)

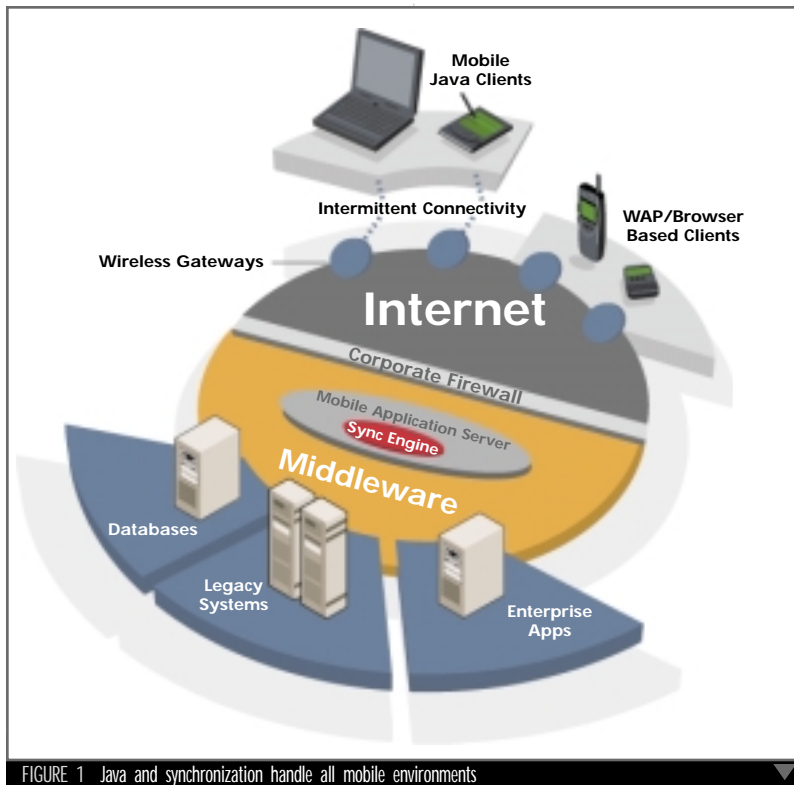


FIGURE 1 Java and synchronization handle all mobile environments

matter which type of databases or applications need to be synchronized, or which client devices need to be supported. Any database, application, or client device can synchronize at the data access-layer level by communicating with industry-standard interfaces, including JDBC, EJB, XML, JCA, JMS, and HTML. You can extend any J2EE services existing on the server to a mobile client device.

### Architecting a Java Synchronization Solution

Conflict resolution during synchronization usually depends on complex business rules that are set by the user and require a robust solution to appropriately handle these rules. Such a solution should support the following:

- **Multiple Device Support**

A robust synchronization solution should be functional on multiple client devices that support offline deployment. J2ME is only one type of client technology that needs to be supported. Other platforms such as PersonalJava (pJava), J2SE, and WABA can also be supported by your wireless platform without requiring additional or third-party add-on products.

Should your synchronization solution connect directly to the client application or require a client database, you'll need to provide for cross-device functionality. The best way to ensure this functionality is to connect your syn-

chronization solution to the data-access layer of the client application. This method allows your synchronization solution to remain flexible in light of the various client devices, platforms, and applications.

- **Real-Time and Asynchronous Access**

Synchronization should not be a standalone product, feature, or function. Your mobile solution should support both real-time (e.g., browser-based) and offline access (e.g., client application and synchronization) to corporate back-end applications and content. Find a vendor who can provide both types of products – otherwise you'll be left with integration and incompatibility issues between your connected and disconnected solutions.

- **Client-Side Data Caching**

Mobile devices should not require a client database for your synchronization solution to work. Many devices don't even have a file system that supports a client database (e.g., Compaq iPaq). If your synchronization solution does depend on a client database, it will require you to add more software products than necessary, raising incompatibility and memory-limitation issues on your client device.

Using the client's cache is a more practical solution. Given that the device will have local memory capacity, you can achieve persistence using this mem-

ory. The memory used for persistence is also less than that used by a client database. Having made the argument for using a caching solution, it's still important that your synchronization solution supports the use of a client database should it be needed due to the nature of the work or the application.

- **Multiple Enterprise Connections**

Your synchronization solution should be able to interface with multiple server-side applications and databases. Avoid using synchronization solutions directly from enterprise application providers; they may work with that particular application, but are not likely to interface well with other back-end content. Find a solution that's open and allows you to interface with multiple back-end solutions; this will save you from significant upgrade and maintenance headaches. You'll need to support connections to JDBC, EJB, XML, JCA, JMS, and HTML data sources, among others.

- **Open APIs**

Find a mobile solution that allows you to develop using API-level access. This development approach enables you to easily integrate with third-party software applications and preserves the flexibility of your current and future mobile strategy.

- **Open Application Development and Java Support**

Ensure that the applications you develop, on both the server- and client-side, use open architectures and industry-standard development methodologies. The most fitting solution for this paradigm is J2EE on the server-side, and J2ME/J2SE on the client-side. If you already have Java in the enterprise, your mobile solution should extend this investment to client devices. Look for a solution that allows you to extend your existing J2EE components or services to mobile devices without the need to rearchitect your existing enterprise IT infrastructure.

- **Flexible Business Logic and Synchronization Rules**

Conflict resolution rules should always take into account the intended functionality of the application. Such synchronization rules can't take a "one size fits all" approach to resolving conflicting data information. In most cases, "last in wins" conflict resolution can have a disastrous effect on the application data accuracy. Synchronization rules should be flexible enough to sup-

# ITtoolbox

[www.ittoolbox.com](http://www.ittoolbox.com)

# Deciding on a synchronization solution is not much different than deciding on a real-time wireless solution

port the various business logic rules of the application, allowing developers and/or users to decide which priority settings should be used if a data conflict occurs.

- **Logging Tables**

Whatever the outcome of the conflict, log tables are necessary to document the results. They must be reliable and easily accessible from within various administrative environments, displaying results as hierarchical tree views of log data, for example. Log tables ensure that even when a conflict occurs, none of the data is lost and the subsequent rollback of the initial conflict resolution can occur.

- **Security**

Allowing mobile devices access to your corporate networks produces security nightmares among your IT staff. Make sure your mobile strategy incorporates industry-standard security technology to encrypt the connection between the mobile device and your corporate networks.

## Understanding Development and Implementation Issues

J2ME is only one type of client technology that allows for disconnected applications. Due to the variety of mobile devices, different technologies emerged to take advantage of the differences in functionalities offered by each device. The two prevalent technologies, both of which happen to be Java-related, are J2ME and PersonalJava (or pJava).

J2ME is mostly used on mobile phones and requires a micro Java Virtual Machine. While the proliferation of J2ME phones is currently very low, future mobile phones are expected to incorporate this technology. Nokia, for example, is currently making a big push

into shipping J2ME phones worldwide.

PJava is generally used on Pocket PCs and takes advantage of the underlying functionality of the operating system. It can deliver a better user experience than a cell phone, due to the additional software and hardware functionality of the mobile device. It appears unlikely that these two technologies will merge anytime in the near future. In fact, it's more likely that new client technologies will emerge, highlighting the need for a robust mobile and synchronization solution that will take these future changes into consideration through open, flexible, and extensible architectures.

Deciding on a synchronization solution is not much different than deciding on a real-time wireless solution. The solution should be based on open standards, leverage existing investment in technologies, and be flexible enough to accommodate future changes in technologies or business direction.

There are many add-on synchronization solutions that may work with your existing enterprise and mobile initiatives. Add-on solutions, however, don't provide the most flexible and robust synchronization as compared to integrated solutions. An integrated solution should provide both real-time and offline functionality of your mobile project. Synchronization should be a simple extension of your real-time wireless solution and shouldn't require a separate technology or software/services vendor.

Whether add-on or integrated, the synchronization solution must be based on open standards and remain flexible for future changes in technologies, not just synchronization technology. An open solution is something that can be managed without the help of the vendor or specialized systems integrator. As

with any software solution, you should not be dependent on the company that produced or sold the software to you, but be able to manage the solution with existing in-house resources.

## Conclusion

When selecting a synchronization solution, many factors play an important role in choosing the solution that's appropriate for the task at hand. Choosing the best-of-breed synchronization solution may not turn out to be the correct choice when you consider other factors. You should make your decision based on a wide variety of influencing issues that are not limited to just your mobile strategy. First, determine the business requirements of your project. Will a wireless solution without offline capability suffice? If not, do you need a mobile solution that can operate in a disconnected mode? Do you need both, allowing certain parts of your organization to use one or the other?

Second, know your existing enterprise and client technology, including both your software and hardware investments. Do you plan on building a new mobile project from scratch? Do you plan on extending an existing enterprise application? Have you invested in Java technology? Are you looking to integrate the wireless/mobile solution into your existing IT infrastructure? Are you looking to build out a completely separate wireless/mobile island, away from the corporate IT system? What type of devices do you plan on supporting – just one or many? Are you planning on mobile-enabling just one or many enterprise applications? Do you need direct access to database content from mobile devices?

Third, evaluate the vendors. Choose one that fits your needs. The more openness and flexibility you (or your vendor) preserves, the better. Choose a product vendor, not a services vendor. Look for a vendor who focuses on and has experience with mobile technologies. Consider the total long-term cost of ownership of your mobile solution, not just the products and/or services cost in the first year.

Synchronization solutions are a must-have for the mobile professional. The current limitations in wireless connectivity require client applications to function even in an offline mode of operation to enhance the productivity of the end user. Synchronization solutions should therefore be an integral part of your mobile strategy. ●

▼▼ [jcapone@aligo.com](mailto:jcapone@aligo.com)

### AUTHOR BIO

Jeff Capone, PhD, is CTO of Aligo, where he leads the technology development and is principal architect of the M-1 Server. He has spent the past decade researching wireless applications.



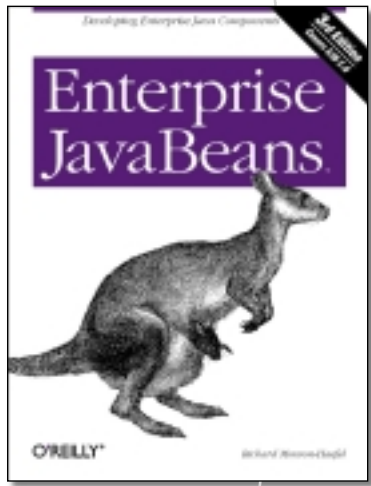
# **Oracle Development Tools User Group**

[www.odtug.com](http://www.odtug.com)

# **Wireless Subscription**

[www.wbt2.com](http://www.wbt2.com)

# Enterprise JavaBeans



I was very impressed with the organization of the book, which made it a treat to read. One of its main strengths is that it stays focused on specific topics and cleanly separates the 1.1 and 2.0 versions of the EJB specification. For example, if you're only concerned with designing a new application using EJB 2.0, the information isn't cluttered with recurring references to the EJB 1.1 CMP model, which is discussed in a separate chapter.

Chapter 1 introduces distributed computing. Chapter 2 provides an excellent overview of EJBs, including coverage of the standard classes and interfaces, the types of beans, and the deployment descriptor for deploying beans. The author has picked a generic application to illustrate his examples – a reservation for a cruise. If this chapter had contained an overall diagram illustrating the entire hierarchy, including the objects developed throughout the book, it would have been a perfect chapter on EJB architecture.

Chapter 3 deals with the basic services provided by the J2EE component transaction monitors or EJB containers. It provides a good discussion on such topics as the EJB life cycle, object persistence, and the bean-container contract. Chapter 4 walks you through developing and deploying some basic EJBs and covers application development concepts from object design to basic database table design. Chapter 5 covers the design of a basic client to access the beans developed in Chapter 4. The discussion on local versus remote interfaces is very helpful.

Chapters 5–8 were, for me, the most useful chapters in the book. Monson-Haefel walks you through the nuts and bolts of designing entity beans with the new EJB 2.0 persistence model. Chapters 6–7 develop sample entity beans in light of the new persistence model with very lucid examples. The organization of the text is excellent. Monson-Haefel begins each example with the abstract programming model, followed by the abstract persistence schema, then the design of the bean interfaces and classes, and finally the deployment of the bean. Each discussion is autonomous and very clear.

Chapter 7 offers clear guidance on database relationships as they relate to EJB 2.0 CMP. Chapter 8 is a good reference for using EJB-QL.

Chapter 9 deals with EJB 1.1 CMP. Chapters 10–11 discuss the details of bean-managed persistence and the EJB container. Chapter 11 has ample coverage of primary keys, the entity context, and the life cycle of entity beans.

Chapters 12–13 cover the other two types of EJBs (session and message-driven beans).

The sample application is developed further in these chapters by TravelAgent Bean, Reservation Process Bean, and associated workflows. Transaction design for EJBs is covered in Chapter 14.

Chapter 15 provides some excellent design strategies for EJB design, such as using hash codes to generate primary keys and dependent value objects to pass objects to and from entity beans. Chapter 16 is a reference chapter on the deployment descriptor for EJBs. This is followed in Chapter 17 by an overview of how EJBs fit into the big J2EE picture. The appendices provide concise references to the EJB APIs and the state and sequence diagrams for the different types of EJBs.

This is one of the best sources of information on EJBs that I've found. It would have been helpful if there were some diagrams of the overall picture, but all in all, this is a very well-organized book about using Enterprise JavaBeans to develop applications.

Table of Contents	
Preface	11. The Entity-Container Contract
1. Introduction	The Primary Key
Setting the Stage	The Callback Methods
Enterprise JavaBeans Defined	EJB 2.0: ejbHome( )
Distributed Object Architectures	EntityContext
Component Models	The Life Cycle of an Entity Bean
Component Transaction Monitors	12. Session Beans
CTMs and Server-Side Component Models	The Stateless Session Bean
Titan Cruises: An Imaginary Business	The Life Cycle of a Stateless Session Bean
What's Next?	The Stateful Session Bean
2. Architectural Overview	The Life Cycle of a Stateful Session Bean
The Enterprise Bean Component	13. Message-Driven Beans
Using Enterprise Beans	JMS as a Resource
The Bean-Container Contract Summary	Message-Driven Beans
3. Resource Management and the Primary Services	14. Transactions
Resource Management	ACID Transactions
Primary Services	Declarative Transaction Management
What's Next?	Isolation and Database Locking
4. Developing Your First Enterprise Beans	Nontransactional Beans
Choosing and Setting Up an EJB Server	Explicit Transaction Management
Developing an Entity Bean	Exceptions and Transactions
Developing a Session Bean	Transactional Stateful Session Beans
5. The Client View	15. Design Strategies
Locating Beans with JNDI	Hash Codes in Compound Primary Keys
The Remote Client API	Passing Objects by Value
EJB 2.0: The Local Client API	Improved Performance with Session Beans
6. EJB 2.0 CMP: Basic Persistence	Bean Adapters
Overview	Implementing a Common Interface
The Customer EJB	Entity Beans Without Create Methods
Persistence Fields	EJB 1.1: Object-to-Relational Mapping Tools
Dependent Value Classes	Avoid Emulating Entity Beans with Session Beans
Relationship Fields	Direct Database Access from Session Beans
7. EJB 2.0 CMP: Entity Relationships	Avoid Chaining Stateful Session Beans
The Seven Relationship Types	16. XML Deployment Descriptors
8. EJB 2.0 CMP: EJB QL	What Is an XML Deployment Descriptor?
Declaring EJB QL	The Contents of a Deployment Descriptor
The Query Methods	The Document Header
EJB QL Examples	The Descriptor's Body
Problems with EJB QL	Describing Enterprise Beans
9. EJB 1.1 CMP	EJB 2.0: Describing Relationships
A Note for EJB 2.0 Readers	Describing Bean Assembly
Overview for EJB 1.1 Readers	The ejb-jar File
Container-Managed Persistence	17. Java 2, Enterprise Edition
10. Bean-Managed Persistence	Servlets
The Remote Interface	JavaServer Pages
The Remote Home Interface	Web Components and EJB
The Primary Key	J2EE Fills in the Gaps
The ShipBean	Fitting the Pieces Together
Obtaining a Resource Connection	Future Enhancements
Exception Handling	A. The Enterprise JavaBeans API
The ejbCreate( ) Method	B. State and Sequence Diagrams
The ejbLoad( ) and ejbStore( ) Methods	C. EJB Vendors
The ejbRemove( ) Method	Index
The ejbFind( ) Methods	
The Deployment Descriptor	

## Enterprise JavaBeans

Author: Richard Monson-Haefel

Publisher: O'Reilly & Associates

Although I've been following EJB 2.0 very closely, it was only recently that I walked into a project that was the perfect venue for its new features, such as the much enhanced container-managed persistence and local interfaces. And *Enterprise JavaBeans*, written by Richard Monson-Haefel, fit the bill as a reference and learning guide.

This is a "must-have" book if you want an introduction to EJBs, are migrating from EJB 1.1 to 2.0, or want to build a new application using EJBs. The author provides a brief introduction to distributed objects and component models in the first chapter; however, if you're unfamiliar with these concepts or with database design, this is not the right book for you. *Enterprise JavaBeans* talks in detail about the obvious – Enterprise JavaBeans – and stays focused on the subject.

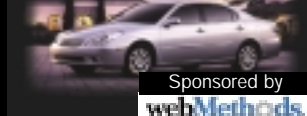
# LARGEST GATHERING OF DEVELOPERS, PROGRAMMERS, AND I-TECHNOLOGY PROFESSIONALS IN THE WORLD!!

**JUNE 24-27, 2002 • JACOB K. JAVITS CONVENTION CENTER • NEW YORK, NY**

**WIN A**

**\$35,000  
LUXURY CAR!**

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!



Sponsored by  
**webMethods**

**REGISTER BY MAY 31  
TO SAVE \$200!  
GO TO [WWW.SYS-CON.COM](http://WWW.SYS-CON.COM) NOW!**

# JAVA XML WEB SERVICES

## Focus on Java

Java, now mainstream, is the dominant back-end technology upon which next-generation technologies are evolving.

Hear from the leading minds in Java how this essential technology offers robust solutions to I-technology professionals and senior IT/IS strategy decision-makers.

**JAVA  
IN  
JUNE  
ESPECIALLY  
IN NEW YORK**



## Java Track: June 25-27

The Java Track, with something for every developer from beginner to advanced, will also look into the role that Java is playing in building up Web Services.

- |  |  |
|--|--|
| <b>JV1</b> Java Security Advanced Concepts   | <b>JV6</b> .NET vs. J2EE                                       |
| <b>JV2</b> Optimizing Database Performance in J2EE Applications  | <b>JV7</b> Building Scalable Web Applications and Web Services |
| <b>JV3</b> Detecting, Diagnosing, and Overcoming the Five Most Common J2EE Application Performance Obstacles | <b>JV8</b> Hot Breaking Session                                |
| <b>JV4</b> Building Asynchronous Applications Using Java Messaging   | <b>JV9</b> Java Tools for Extreme Programming                  |
| <b>JV5</b> Building "Smart Client" Applications using J2SE and J2ME  | <b>JV10</b> Building Truly Portable J2EE Applications          |
|  | <b>JV11</b> Security for J2EE Application Servers              |

## Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

**XML  
NEXT G  
OF  
ENTERPRISE  
DEPLOYMENT**



## XML Track: June 25-27

The XML Track will focus on the various facets of XML technologies as they apply to solving business computing problems. This track targets audiences ranging from beginners to system architects and advanced developers.

- |   |  |
|---|--|
| <b>XM1</b> Data - a Key Part of Web Services  | <b>XM7</b> XML Web Services: Standards Update                |
| <b>XM2</b> OASIS Standards Update   | <b>XM8</b> Bringing XML to PKI                               |
| <b>XM3</b> A Universal Business Language  | <b>XM9</b> Building a Corporate Information Model in XML     |
| <b>XM4</b> Achieving Standards-Based Mobile eBusiness Success with XML & Web Services   | <b>XM10</b> XML in the Enterprise and Inter-Enterprise World |
| <b>XM5</b> Using XML for Rapid Application Development and Deployment with Web Services | <b>XM11</b> XML and RDB Relationships                        |
| <b>XM6</b> Open Up Your RDBMS with Open Standard Technologies                           |  |

## Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at Web Services Edge East 2002.

Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting-edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Keynote panel that will debate and discuss the realities and promise of Web Services.

**WEB  
SERVICES  
SKILLS,  
STRATEGY,  
AND  
VISION**



## Web Services Track: June 25-27

The Web Services Track will concentrate on the latest emerging standards. It will include discussions of .NET, Sun ONE, J2EE and App Servers, the role of UDDI, progress of the standards-making bodies, SOAP, and Business Process Management. It is intended for developers, architects, and IT management.

- |   |  |
|---|--|
| <b>WS1</b> Starting Out In Web Services: Fundamentals In Web Services   | <b>WS7</b> Developments in Web Services Standards            |
| <b>WS2</b> State of the Web Services Industry                           | <b>WS8</b> Unlocking the Value of Enterprise Web Services    |
| <b>WS3</b> Web Scripting Languages: Options for Dynamic Web Development | <b>WS9</b> Guarding the Castle: Security and Web Services    |
| <b>WS4</b> Building a Web Services Application Infrastructure           | <b>WS10</b> Practical Experiences with Web Services and J2EE |
| <b>WS5</b> Deploying a Corporate Portal                                 | <b>WS11</b> Designing Web Services Using UML                 |
| <b>WS6</b> The Enterprise Service Bus (ESB): Leveraging Web Services    |  |

## .NET TRACK: June 25-27

Combining technology, platform, and architecture, the .NET Track provides insight on the latest developments for the Windows platform.

- |  |
|--|
| <b>NT1</b> Configuring .NET for Security, Performance, and Reliability |
| <b>NT2</b> Changing Your Environment to .NET                           |
| <b>NT3</b> Going Mobile with .NET                                      |
| <b>NT4</b> .NET on Other Platforms (FreeBSD, MONO, Portable .NET)      |
| <b>NT5</b> Inside the CLR  |
| <b>NT6</b> Accessing Data from .NET                                    |
| <b>NT7</b> Advanced .NET Web Services                                  |
| <b>NT8</b> Migrating Legacy Code to .NET                               |
| <b>NT9</b> Advanced User Interfaces with GDI+                          |

## IT Strategy Track: June 25-27

The IT Strategy Track will focus on the managerial and design aspects of the various development disciplines. Topics will include Standards, Architecture, Design Patterns and Best Practices, Project Planning and Management, and Extreme Programming.

- |   |  |
|---|--|
| <b>IT1</b> Developing, Deploying and Managing Web Services                                  | <b>IT8</b> Minimizing Risks and Maximizing Investments in J2EE Development Through the Use of Reusable Application Architecture and Frameworks |
| <b>IT2</b> Key Trends and Technologies for Building an Enterprise Web Services Architecture | <b>IT9</b> Application Integration - Building a Flexible Web Services Architecture   |
| <b>IT3</b> Selecting a Framework: Toolkit, Platform, or Roll Your Own?                      | <b>IT10</b> The Economics of Web Services  |
| <b>IT4</b> Getting Started with Web Services  | <b>IT11</b> Hooking It All Together - Application Integration Case Study   |
| <b>IT6</b> Overcoming the Web Services Barriers   |  |
| <b>IT7</b> The Real Issue: Improving Your Enterprise with Enterprise Web Services           |  |

## Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- I-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

## Sun Microsystems at JDJEdge 2002: Java Fast Paths & Java University<sup>SM</sup> Program at the Jacob K. Javits Convention Center, NYC

### Java Fast Paths

Attend fast-paced, code-level, full-day Java technology developer training designed to provide you with the skills to confidently approach the industry's core Java technology certification exams. Don't just say you know it... prove it! Most developers recognize the expanding importance of gaining Java technology certification. If you're like those who've already taken the exams, the real hurdle to certification is finding time to prepare for the tests.

### Monday, June 24, 2002:

- Java™ 2 Platform: Developer Certification Fast Path
- Java™ 2 Platform: Architect Certification Fast Path
- Web Component Developer: Certification Fast Path



### Java University<sup>SM</sup>

The Java University<sup>SM</sup> Program complements this year's JDJEdge Conference by offering three aggressive, full-day, code-level training classes for experienced software developers, architects and engineers.

Attend code-level training in sessions designed by industry luminaries, and recognized experts. Sessions cover XML and Web services technology. Whether you're a beginning or a veteran developer, architect, or software engineer, you'll benefit from these value-packed full-day courses. Register now. Seating is limited.

### Tuesday, June 25, 2002

Developing Solutions Using Java™ Technology and XML

### Wednesday, June 26, 2002

Web Services Programming Using Java™ Technology and XML

### Thursday, June 27, 2002

Java™ APIs for Enterprise Web Services

# Java-Miner

by CAST

REVIEWED BY JASON BELL [jasonbell@sys-con.com](mailto:jasonbell@sys-con.com)

## CAST

3, rue Marcel Allégot  
92 190 Meudon  
Paris, France

Phone: +33 1 46 90 21 00

Fax: +33 1 46 90 21 01

Web: [www.castsoftware.com](http://www.castsoftware.com)

## Test Environment

Computer: Compaq Presario 1920

Processor: 300MHz Pentium III

Hard Drive: 4GB

Memory: 64MB

Platform: Windows ME

## Specifications

Platforms: Windows 98, 2000,  
or NT 4.0

It's unfortunate that programmers come and go at an alarming rate in the IT industry, leaving code that must be maintained by someone who quite frequently had no hand in writing it. Software engineers using UML have models on how their programs behave, but the rest of us are left to read through reams and reams of methods. Most of the time all you need is an overview of how the program is made up, not the finer details.

### Java-Miner

CAST's Java-Miner provides a graphical "roadmap" of Java packages, classes, and their associated methods and constructors. As the saying goes, "A picture paints a thousand words"; this application can tell the whole story behind Java classes in a way that's easy to interpret.

### Installing and Using Java-Miner

Installation is straightforward. A self-installing package can be downloaded from CAST's Web site (it's a 15-day evaluation) or it can be installed from the distribution CD. The Help documentation is also installed onto your system, so you can use Help and program simultaneously.

Java-Miner works via a series of wizards that guide you through the process of loading in a Java application to view. When starting the program you're asked whether you want to load an existing analysis or create a new one. Creating a new analysis is a simple process, again using a wizard to gather the required information. A project name is required, which becomes the XML output filename. This, by default, is saved to the root directory of the "C" drive on your system, so it's very important to use the directory-browsing function to place the output files somewhere sensible. In my opinion, this is a big oversight and perhaps a default directory should be included somewhere in the proceedings. Within this wizard the Java source files and associated classpaths are added. Once the

"Start" button is clicked, Java-Miner will analyze the packages and create the roadmap.

An object browser provides a dropdown view of the analysis you're working on. Many IDEs work this way so most people will easily make sense of this. The graphical layout is basically a large blank workspace that's scrollable. Within this area you can zoom in and focus on specific components or zoom out for an overall picture. These graphical views can be output to a printer for a permanent copy of the analysis.

Working with the analysis is easy, but the blank screen is a little off-putting to the new user. I expected to see some sort of roadmap on-screen, which could be customized as required. Instead the user is expected to drag the object onto the output screen, which drops the object onto the roadmap. It's up to the user to expand the other methods and constructors of that object by right-clicking and then selecting "Expand." The parts of the object will appear and link together. The expanded links provide a wealth of information about where methods are called from and which objects they belong to (see Figure 1).

For a programmer looking at some code for the first time, it's an enormous benefit to be able to learn the program structure without reading through pages and pages of source code.

With large applications the amount of detail on the roadmap can be overwhelming, so you can tailor the amount of detail displayed by setting up a series of layers. For example, you can create a layer to display the relationship links between the methods and another layer to display the objects, not the methods within them. Another useful feature is Java-Miner's ability to work its way through a batch of objects and show a roadmap from one specified method to another. So if you know that an object references a method but can't see how, let Java-Miner do the work for you.

### Summary

My first impressions were positive, though I was a little surprised about the file locations that Java-Miner was using; with a little planning this won't be a problem for most people. For those who need to maintain a working application in which time is precious, Java-Miner may be the answer. ☛

### Product Snapshot

Target Audience: Java programmers, business analysts

Level: Beginner to advanced

#### Pros:

- Easy to use
- Graphical roadmaps are easy to read.

#### Cons:

- No default directory; used defaults to the root directory

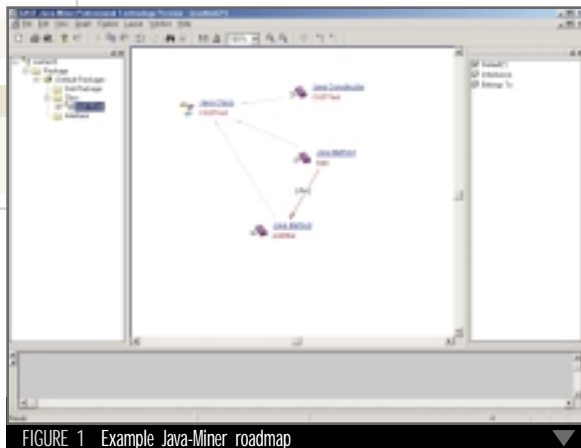


FIGURE 1 Example Java-Miner roadmap

# Web Services Edge Conference & Expo

[www.sys-con.com](http://www.sys-con.com)



# Manifest Destiny

Simplify the packaging and releasing of Java applications

WRITTEN BY  
NORMAN  
RICHARDS

**R**eleasing Java applications can be a real challenge. Fortunately, Java provides a rich set of features for packaging and deploying applications that can simplify the release process significantly.

This article presents some of the issues involved with packaging Java code. I'll explore the Java manifest file and suggest ways it can be used to manage JAR file dependencies and to eliminate classpath issues normally associated with cross-platform deployment. I'll also explain how to use the package-versioning features of the manifest to ensure the compatibility of packages used.

## What Is a JAR File?

During development, Java classes are normally compiled into local directories on a disk. Java applications can be run and even distributed in this manner, but it's not a practical way to work. Fortunately, a more manageable approach is available. Java class files can be packaged in Java Archive (JAR) files for distribution and execution.

A JAR file is actually just a ZIP archive of the class files. Using this well-known archive format makes JAR files exceptionally easy to work with. Many tools and libraries exist for manipulating ZIP files, so for the developer the tool set is very rich. However, since they're in a simple archive format, JAR files can't natively express meta-information about the applications they contain.

## The Manifest Is Born

To provide meta-information that describes the archive, the JAR file designates a special directory to hold the meta-information, the META-INF directory. For our purposes, I'm concerned with only one file in this directory, MANIFEST.MF. This is the manifest file for the JAR. It describes the contents of the JAR file and provides application information to the JVM at runtime. Most JAR

files contain a very basic manifest file by default. Try examining the META-INF/MANIFEST.MF file using any JAR (or ZIP) program. This can be done with any ZIP file tool or by using the JAR command directly.

```
jar xvf myapplication.jar META-INF/MANIFEST.MF
```

If your JAR file was created by the JAR tool, you should see a simple default manifest file.

```
Manifest-Version: 1.0
Created-By: 1.4.0-beta(Sun Microsystems Inc.)
```

This trivial manifest file lets me know I'm working with a version 1.0 manifest file. Right now this is the only defined manifest file format. The next line tells me this JAR was created by the JAR utility in the Sun Java 1.4 beta SDK. If the manifest file was created, for example, by the Ant build tool, then you might expect to see something like "Created-By: Ant 1.2". When creating your own manifest files, put in text that's relevant to your own project.

## The Basic Format

The format of the manifest is simple. Each line of the manifest file is a name-value pair. The attribute name is first, followed by a colon, and then the attribute value. Lines should be limited to 72 characters, but if you need more than that, you can continue a value on the next line by starting the line with a space. Any line that begins with a space is considered a continuation of the previous line.

All the attributes at the top of the file are global. You can also apply an attri-

bute to a specific class or package. I'll show examples of that later.

## Inserting the Manifest

To add a manifest file to a JAR file, the "m" option is used and the filename of manifest file is passed to the JAR program after the name of the JAR file.

```
jar cvfm myapplication.jar myapplication.mf -C classdir .
```

If you use the Ant build tool, the manifest can be specified by adding a manifest attribute to the JAR specification:

```
<target name="jar">
  <jar jarfile
    ="myapplication.jar"
    manifest="myapplication.mf">
    <fileset dir="classdir"
      includes="**/*.class"
    />
  </jar>
</target>
```

## A Java Executable

Now that you've had a small taste of the manifest, let's step back and look at a few deployment issues that can be greatly simplified by the manifest. To launch a standard Java application, I need to invoke a Java Virtual Machine and include the application JAR in the classpath, then specify the class I want to invoke. Let's assume again that the JAR is myapplication.jar and the application main class is com.example.myapp.MyAppMain. The application start script will need to invoke:

```
java -classpath myapplication.jar
com.example.myapp.MyAppMain
```

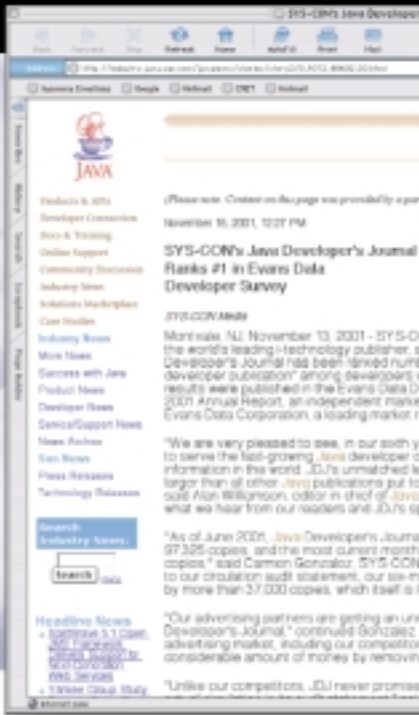
**AUTHOR BIO**  
Norman Richards works at Commerce One Labs, the research division of Commerce One.

# **SYS-CON Industry Newsletters**

[www.sys-con.com](http://www.sys-con.com)



# Q: Do you know why *Java Developer's Journal* is the world's leading Java resource?



## A:

- Because JDJ covers issues of interest to the entire Java community (in excess of 500,000 i-technology professionals) and decision makers.
- JDJ has been named the "most trusted magazine" in the industry.
- JDJ is edited by Alan Williamson and the industry's most respected editorial advisory board.
- Most important, JDJ is read by professionals like you.

**JAVA DEVELOPERS JOURNAL**

While this isn't an overwhelmingly complicated task, it's awkward specifying the main class external to the JAR file. If I were writing a stand-alone application, I'd expect the application to know the class I want to start execution at. I can specify this with the Main-Class attribute in our manifest.

```
Manifest-Version: 1.0
Created-By: JDJ example
Main-Class: com.example.myapplication
```

Now, if I add the manifest to our JAR, I can invoke our Java application much more simply.

```
java -jar myapplication.jar
```

This is definitely a much simpler and less error-prone way to launch an application.

### Managing JAR Dependencies

This first step is good, but very few Java applications can be distributed as a single JAR. Typically, I need support libraries. Suppose my application is using Sun's Javamail classes and I need to include activation.jar and mail.jar in my classpath. My previously simple Java -jar command becomes a bit less pleasant.

```
java -classpath mail.jar:activation.jar -jar myapplication.jar
```

Matters are further complicated by the fact that the classpath specification varies between operating systems. On Unix, classpath elements are separated by a colon, but on Windows, classpath elements are separated by a semicolon.

Again, the manifest file provides a way for me to manage this complexity. What I really have is a JAR dependency. myapplication.jar now depends on mail.jar and activation.jar. Anytime I use myapplication.jar I'll want those two JARs. I can express this dependency in the myapplication.jar manifest.

```
Manifest-Version: 1.0
Created-By: JDJ example
Main-Class: com.example.myapplication
Class-Path: mail.jar activation.jar
```

Now I can once again invoke the application in the original simple manner.

```
java -jar myapplication.jar
```

Let's look more closely at how this works. The Class-Path attribute is a space-separated list of relative URLs pointing to JAR files that the current JAR references. Keep in mind that I need to escape spaces and special characters as I would in a URL (a space would be escaped as "%20"). I also need to use a forward slash ("/") as the directory separator, regardless of the platform preference. Also note that the URL reference is indeed relative. I couldn't specify C:/some.jar or /usr/local/java/lib/someother.jar as a dependency if my intent is to root my file search path anywhere other than the current directory.

Also, note that when I install the application I still need to make sure that mail.jar and activation.jar are in the same directory as myapplication.jar, otherwise the application will fail to find the JARs I need.

As an alternative, some developers prefer to place support JARs in a sub-directory. For example, suppose that next to the application JAR is a directory ext that contains the support libraries. I could use a manifest like this:

# Simplex Knowledge Company

[www.skcn.com](http://www.skcn.com)

# Q: Do you know that all ads that appear in JDJ must be pre-approved by the publisher before being accepted?

## A:

- Although every Java vendor wants to advertise in JDJ, we set high standards for accepting ads.
- You may see some vendors advertising elsewhere, having failed to meet our strict qualifications.
- Companies that don't qualify to advertise in JDJ search for alternate advertising venues.

Actavis Corporation  
 Altek Corporation  
 Alveo  
 AltiHub  
 AppDev Training Company  
 Appinity  
 Aquatica Solutions  
 Ashworth Corporation  
 BEA  
 Blaze Advisor From BMC  
 Barland Software Corp.  
 Casoo Engineering AG  
 Capella University  
 Compas Software  
 Compware Corp.  
 Corda Technologies  
 DataDirect Technologies  
 Dica.com  
 ESI  
 eSolen Corporation  
 Firana Software  
 Hewlett-Packard Company  
 HT Software  
 IBM  
 ILOG  
 InetSoft Technology Corp.  
 Infragility, Inc.  
 Invision Technology  
 IntelShield Software Corp.  
 INT, Inc.  
 Introware  
 iSparix Corporation  
 John Wiley & Sons  
 L003 Software Inc.  
 Luteis Technologies, Inc.  
 Metroworks Corp.  
 Mopson Technology  
 New Atlanta Communications  
 Northwoods Software Corporation  
 Oracle Corporation  
 ParSoft Corporation  
 PointBase, Inc.  
 Prosoft Technologies  
 Pratica Software  
 prafimative Solutions  
 Praxia Hall PDR  
 QUADRAM Incorporated  
 Quintessence Systems Limited  
 Rational Software  
 ReportMill Software, Inc.  
 Sharp  
 SilverStream Software  
 Simplex Knowledge Company  
 Sitaka Software  
 Software, Inc.  
 Sonic Software  
 SpiritSoft  
 SOLUTIONSgate  
 Sun Microsystems  
 Sun Solutions  
 Syntex  
 Thought Inc.  
 ToplineSoft Corporation  
 Visual Mining  
 WebGain, Inc.  
 Zoro G

**JAVA** DEVELOPERS JOURNAL

```
Manifest-Version: 1.0
Created-By: JDJ example
Main-Class: com.example.myapplication.MyAppMain
Class-Path: ext/mail.jar ext/activation.jar
```

Again, keep in mind that the classpath components need to be specified as if they were a relative URL component and were not using platform-specific naming conventions.

## Multiple Main Classes

It's not uncommon for Java applications to provide multiple entry points. Perhaps you have both a command-line and a GUI version of your application, or perhaps a main application with several support tools that share a lot of the same code base.

In this case, a good approach is to package all the common JARs into a single application code JAR. Separate application JARs could be created for each entry point, each containing a manifest file with a dependency on the library JAR and a main-class entry that represents the entry point. Depending on the development setup, you may find it simpler to keep all the classes (including the front-end classes) in the library JAR or to package the front-end classes only in their corresponding JAR. The latter is cleaner, but may be troublesome if you haven't organized your classes for easy separation.

The resulting application would look like the following: the lib JAR contains the class files and the dependencies for those class files; the remaining JARs are empty, minus the manifest, and depend only on the classes JAR.

```
Manifest for.myapplicationlib.jar:
Manifest-Version: 1.0
Created-By: JDJ example
Class-Path: mail.jar activation.jar
```

```
Manifest for.myapplicationconsole.jar:
Manifest-Version: 1.0
Created-By: JDJ example
Class-Path:.myapplicationlib.jar
Main-Class: com.example.myapplication.MyAppMain
```

```
Manifest for.myapplicationadmin.jar:
Manifest-Version: 1.0
Created-By: JDJ example
Class-Path:.myapplicationlib.jar
Main-Class: com.example.myapplication.MyAdminTool
```

## Package Versioning

After you've distributed code, a key issue in release management is figuring out exactly what code you have. What version of the code is running now? What versions of the support libraries are you using? There are a number of approaches to this problem, but the manifest file provides a very elegant solution. In the manifest file, you can describe every package provided in the JAR file according to the Java versioning system.

Java is based on the principal of separating the specification of a technology from its implementation. A package's specification defines what the package is and the implementation defines who is providing the implementation of the specification. The specification and implementation are described by a name, a version number, and a vendor. To get a

PROPERTY	VALUE
java.vm.specification.version	1.0
java.vm.specification.name	Java Virtual Machine Specification
java.vm.specification.vendor	Sun Microsystems Inc.

TABLE 1

# Dynamic Buyer Incorporated

[www.ibm.com/small business/dynamicbuyer](http://www.ibm.com/small_business/dynamicbuyer)

# Q: Do you know that the world's leading Java vendors are JDJ advertising partners?

Actuate Corporation  
 Allowits Corporation  
 Altea  
 AltiWeb  
 ApplDev Training Company  
 Applicity  
 Aquatica Solutions  
 Ashworth Corporation  
 BEA  
 Black Advisor from IBM  
 Burland Software Corp.  
 Camoo Engineering AG  
 Capella University  
 Compose Software  
 Compuware Corp.  
 Corda Technologies  
 DataDirect Technologies  
 Dixie.com  
 ETRI  
 eKirlan Corporation  
 Erona Software  
 Hewlett Packard Company  
 HIT Software  
 IBM  
 IBDG  
 InetSoft Technology Corp.  
 Infragistics, Inc.  
 Invention Technology  
 InstillShield Software Corp.  
 INT, Inc.  
 Introware  
 ISovis Corporation  
 John Wiley & Sons  
 L000 Software Inc.  
 Lunit Technologies, Inc.  
 Metworks Corp.  
 Mongoose Technology  
 New Adanta Communications  
 Northwoods Software Corporation  
 Oracle Corporation  
 PicoSoft Corporation  
 PointBase, Inc.  
 Pransel Technologies  
 Precise Software  
 proInnovative Solutions  
 Prorize Hall PTR  
 QUILCORN Incorporated  
 Quintessa Systems Limited  
 Rational Software  
 ReportMill Software, Inc.  
 Sharp  
 SilverStream Software  
 Simplex Knowledge Company  
 Sitnick Software  
 Software, Inc.  
 Sonix Software  
 SpintSoft  
 SOLUTIONSsite  
 Sun Microsystems  
 Sun Solutions  
 Sylace  
 Thought Inc.  
 TogetherSoft Corporation  
 Visual Mining  
 WebGate, Inc.  
 Zero G

## A:

- Our strict elimination process prequalifies JDJ advertising partners.
- JDJ accepts and prints ads from our advertising partners only if we think that the message in these ads might be of direct interest to our readers.
- We also make sure that our advertising partners, and their products and services, have a reputation as leaders in their markets.

**JAVA** DEVELOPERS JOURNAL

feel for this information, let's first look at a few JVM system properties (queryable via `java.lang.System.getProperty()`).

Java first defines the version of the JVM specification being used (see Table 1). This explains that the JVM I'm running is an implementation of Sun's JVM Specification 1.0. It doesn't tell me who created the JVM, it just states that the JVM conforms to a certain standard. To see implementation details, I have to look at the implementation properties. Table 2 is from a Java 1.3 release from Sun.

Again, this states that the JVM is from Sun and is the HotSpot Client JVM. The version is 1.3.0\_04, which happens to correspond with the API version number but is not formally related to it. The Java API specifica-

PROPERTY	VALUE
<code>java.vm.version</code>	1.3.0_04
<code>java.vm.name</code>	Java HotSpot(TM) Client VM
<code>java.vm.vendor</code>	Sun Microsystems Inc.

TABLE 2

tion and implementation are also defined in system properties ("java.specification.version", "java.specification.name"), but since I'm focusing on my own releases, not the JVM, I'll show how to apply Java versioning to the libraries.

In the manifest file, I can define both a specification and an implementation version for each package by declaring a named entity and then adding specification and implementation attributes. These attributes are:

- Specification-Title
- Specification-Version
- Specification-Vendor
- Implementation-Title
- Implementation-Version
- Implementation-Vendor

Specification information is useful primarily when providing a library or programmatic interface to the outside world, or implementing a programmatic interface defined by a standards body. However, another candidate for specification information for an application would be a design or requirements document, if your development process calls for it.

Implementation attributes can be used to track the internal release information. A common technique is to track internal build numbers in the Implementation-Version field.

Let's extend the manifest to include package information. Suppose I'm releasing build 2002-03-05-A of version 2.4 of MyApp; I could represent this with the following manifest:

```

Manifest-Version: 1.0
Created-By: JDJ example
Class-Path: mail.jar activation.jar

Name: com/example/myapp/
Specification-Title: MyApp
Specification-Version: 2.4
Specification-Vendor: example.com
Implementation-Title: com.example.myapp
Implementation-Version: 2002-03-05-A
Implementation-Vendor: example.com
  
```

Remember to include a blank line between the main attributes section and the entity attributes section and to use slashes, not dots, when referring to packages and class-attribute sections. Think of this as referring to the name of the entity as it exists in the JAR file, not by the true package or class names.



# Web Services JOURNAL

Special  
Introductory  
Offer  
SAVE \$13.89\*

## The world's leading independent Web Services information resource

Get Up to Speed with the Fourth Wave in Software Development

- Real-World Web Services: Is It Really XML's Killer App?
- Demystifying ebXML: A Plain-English Introduction
- Authentication, Authorization, and Auditing: Securing Web Services
- Wireless: Enable Your WAP Projects for Web Services
- The Web Services Marketplace: An Overview of Tools, Engines, and Servers
- Building Wireless Applications with Web Services
- How to Develop and Market Your Web Services
- Integrating XML in a Web Services Environment
- Real-World UDDI
- WSDL: Definitions and Network Endpoints
- Implementing SOAP-Compliant Apps
- Deploying EJBs in a Web Services Environment
- Swing-Compliant Web Services
- and much, much more!

Only \$69.99 for 1 year (12 issues)  
Newsstand price \$83.88 for 1 year



**SYS-CON MEDIA**

SYS-CON Media, the world's leading publisher of Technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of Web services.

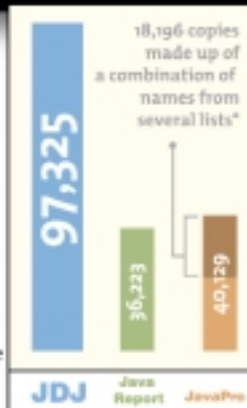
\*Offer subject to change without notice

# Q: Do you know that JDJ's competitor is no longer in print?

Current Circulation: June 2001

Now that our competitor is no longer in print...

...JDJ is here to serve you as your exclusive media partner for many years to come!



\*see BPA statement

## A:

- JDJ's competitor, Java Report, hasn't been printed since October 2001.
- After a strict evaluation process, we are proud to announce that we have decided to accept advertisements from a limited number of qualified vendors who used to advertise in Java Report.
- We thank all our advertising partners for making it possible for us to bring you another great issue of Java Developer's Journal.

**JAVA DEVELOPERS JOURNAL**

It's quite likely that the JAR file will provide multiple packages. They can all be specified in the manifest. Just remember to leave a blank line between each section.

### Querying Package Version

Releasing package information in the manifest is good, but ideally I'd like to be able to access package information at runtime. Java provides convenient access to the manifest package attributes via the `java.lang.Package` class. There are three basic ways to get a package object.

1. **`Package.getPackages()`**: Returns a list of all the packages currently defined by the system
2. **`Package.getPackage(String packagename)`**: Returns a package by name
3. **`Class.getPackage()`**: Returns the package object for a given class

Using these functions, I can provide dynamic logging of package version information (see Listing 1).

It's very important to note that if no classes have been loaded from a given package by the class loader, then there will be no `Package` object available for the package. That's why I test for null at the beginning. This is important because you may be tempted to log all the packages in the system at startup. However, it's unlikely that all the relevant packages in the application will have been loaded yet. Still, package version information can be quite valuable (particularly for support purposes) so it's a good idea to make this information readily available.

### Summary

The manifest file provides a wealth of useful metainformation and greatly simplifies the task of packaging and releasing Java applications. We've seen how to package classes in JAR files, how to track dependencies between JARs, how to turn JARs into executables, and how to track the versions of packages in classes. These are the basic tools for application release management in the Java platform. Making proper use of this functionality can greatly simplify this task. ☛

### Manifest Tips

- Always start a manifest with the `Manifest-Version` attribute.
- Limit lines to 72 characters; if longer lines are needed, use line continuations.
- Make sure the last line has a carriage return at the end; otherwise the line will be ignored.
- If a `Class-Path` entry crosses directories, use `/` as the directory separator, regardless of the platform.
- Separate main attributes from entity attribute sections by a blank line.
- Use slashes, not dots, to delimit package and class entities.
- Class entities should end with `.class` and package entities should end with a slash (`/`).

norman.richards@commerceone.com

#### Listing 1

```
void logPackage()
{
    Package package =
        Package.getPackage("com.example.myapp");
    if (package == null) {
        System.out.println("com.example.myapp not loaded");
        return;
    }
    System.out.println("com.example.myapp version " +
        package.getSpecificationVersion() +
        " build #" +
        package.getImplementationVersion());
}
```

Download the Code!  
www.javaDevelopersJournal.com



# JDJ Store

[www.jdjstore.com](http://www.jdjstore.com)



# WIRELESS AND WEB SERVICES COMBINE TO KEEP JAVAONE JIVING...AND JAVA THRIVING

by JDJ News Desk

**“WHAT WERE YOUR OVERALL IMPRESSIONS OF JAVAONE 2002?”** we asked a select handful of the movers and shakers of contemporary enterprise Java-based computing

(April 12, 2002) – It’s a cliché that, in a difficult economy, it is business leaders and business leaders alone that stand out. But it’s a cliché that was shown to be at least partly valid by what was on display – and who was displaying it – at JavaOne 2002, which took place at the end of March in San Francisco’s huge Moscone Center....

#### *JavaOne 2002 Was “Amazing,” Says Gosling*

James Gosling, the “father” and co-creator of Java and never one to mince words, was unequivocally jazzed by the whole event: “It was amazing,” he says, simply.

“Despite the economy and awkward scheduling,” Gosling continues, “attendance was strong, and everyone I met was charged up. I didn’t encounter anything that felt like the vapor of past years: just solid companies with actual revenue.”

Was there anything that stuck out as being a new killer app of Javaland? “The technology has become so broad that it’s hard to ‘pick a winner’ – in a sense, they all are,” Gosling replies. “To me, it’s this broadness that is the real exciting feature these days: the whole end-to-end architecture is there.”

But once a geek always a geek, and no one who knows Gosling should be the least surprised to hear him confess: “Then, of course, there were those cases at the entrance to the exhibit halls filled with very cool, very lovely gadgets. Major objects of geek lust!”

#### *Web Services and Wireless Are the New Hot Areas*

Bill Roth, Sun’s group marketing manager, is completely candid about the realities of the present economic turbulence.

“My expectations were pretty low,” he says, “given the economy. Most shows have been at about 50% capacity.” But he was impressed, he explains, with the scope and results of JavaOne.

“I was impressed with traffic...which underscores two important items: 1) there are a sufficient number of new folks coming into the Java world to sustain and grow the community, and 2) there is sufficient new content, primarily in the area of Web services and wireless to hold developers’ interest in the long term.” “While I remain a huge fan of J2EE,” Roth concludes, “I am utterly amazed by the non-PC devices running Java.”

#### *Sun’s Own EVP Is “Blown Away”*

Patricia C. Sultz, executive vice president of Sun’s Software Systems Group, is as enthusiastic about JavaOne 2002 as James Gosling is. Her way of explaining it is a little different, though, and makes reference to Java processes as much as to products.

“First, I was blown away by all the different ways that people are using Java and what a huge success it is,” she says. “Java is clearly the de facto standard for server-side development. I continue to be amazed at how adoption keeps growing and more and more applications keep appearing that enable enterprise and legacy data to be accessed with Java. It’s really a testament to what a community working together, like the Java Community Process, can accomplish.

“Second,” Sultz continues, “the momentum of Java technology in the wireless space is really strong. Large wireless carriers around the world realize that interactive applications create a whole new revenue opportunity and that Java provides an independent, standards-based application platform for them. This combined with the exciting range of phones and mobile devices being offered or on the way will drive Java technology’s acceptance to unprecedented levels.”

**RECEIVE \$150**  
DISCOUNT OFF FULL CONFERENCE  
WEB SERVICES EDGE REGISTRATION



# Learn How to Develop SOAP Web Services NOW!

at a One-Day Seminar...Coming to a City Near You!

**Paolini and Williamson Praise Return to Basics**

George Paolini, chief marketing officer of Zaplet, Inc., is in no doubt: "All-in-all a great show," he affirms. "I saw a focus on back to basics, and that's a strong reflection of Java's role in the enterprise, which has been clearly established."

Alan Williamson, who as editor-in-chief of *Java Developer's Journal* has become famous for calling for just such a back-to-basics shift for the past year, agrees.

"This JavaOne was a great one as usual," enthuses Williamson. "Of all the JavaOnes I can remember," he continues, "this was one of the ones that makes this particular event so great." While the overall foot traffic may have been down significantly, he

phase of Web services, the definition phase, is drawing to a close."

"Without exception," he continues, "vendors had already provided the back-end connectivity within their products to allow them to expose a Web service interface." As Rhody succinctly puts it, so far as Web services is concerned, "The battle for acceptance is over. The battle for respectability has begun."

It is a battle that is next joined June 24-27 at Web Services Edge 2002 (East) International Web Services, Java, and XML Conference & Expo at the Jacob Javits Convention Center in New York City ([www.sys-con.com/webservicesedge2002east](http://www.sys-con.com/webservicesedge2002east)).

**“ Java is clearly the de facto standard for server-side development... adoption keeps growing and more and more applications keep appearing that enable enterprise and legacy data to be accessed with Java ”**

adds, there was "time to talk to people and find out what was happening in the real world with Java."

The major points that Williamson draws attention to are first the fact that people seemed to have given up on always-on wireless. "Where in previous years," he says, "companies were building software assuming that a network would be there that was always on, this hasn't transpired, so a lot of those same companies are now concentrating on synching software to make use of the network when and if it's online."

Second, Williamson noticed that while in previous years, "software tool vendors followed the vogue and produced a whole raft of UML tools, intimidating all those who didn't use them, and now the shift has moved to performance and analysis tools."

**"Respectable" Web Services Here We Come?**

Williamson's last point struck a chord, not surprisingly, with the editor-in-chief of *Web Services Journal*, Sean Rhody, who declares that it was clear to him at JavaOne 2002 that "the first

**Last Word to Scott McNealy**

No wrap-up of JavaOne would be complete without some kind of 65,000-foot view from Sun's highest flier, chief executive Scott McNealy. Asked by *JDJ* what his strongest impressions of the week were, from a Java industry point of view, he cuts straight to the chase: "To me, it all comes back to open - and open means interoperability. That's the promise of Java, and we all have to do our best to keep that promise, year after year."

"Every chance I get," McNealy continues, "I urge developers to test their applications for compatibility on at least two app servers to make sure they haven't picked up any proprietary extensions in the tools they use. Incompatibility is the wrong answer. It limits the value of your application and the size of your market."

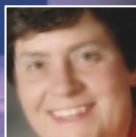
Now who in Javaland could possibly quarrel with that? ☛

To read the full text, go to [www.sys-con.com/java/](http://www.sys-con.com/java/)

# Jump-start your Web Services knowledge. Get ready for Web Services Edge East and West!

AIMED AT THE JAVA DEVELOPER COMMUNITY AND DESIGNED TO EQUIP ATTENDEES WITH ALL THE TOOLS AND INFORMATION TO BEGIN IMMEDIATELY CREATING, DEPLOYING, AND USING WEB SERVICES.

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



**PRESENTERS...**

**Anne Thomas Manes, Systinet CTO**, is a widely recognized industry expert who has published extensively on Web Services and service-based computing. She is a participant on standards development efforts at JCP, W3C, and UDDI, and was recently listed among the Power 100 IT Leaders by Enterprise Systems, which praised her "uncanny ability to apply technology to create new solutions."



**Zdenek Svoboda is a Lead Architect** for Systinet's WASP Web Services platform and has worked for various companies designing and developing Java and XML-based products.

EXCLUSIVELY SPONSORED BY



**BOSTON, MA** (Boston Marriott Newton) **SOLD OUT!** **JANUARY 29**  
**WASHINGTON, DC** (Tysons Corner Marriott) **SOLD OUT!** **FEBRUARY 26**  
**NEW YORK, NY** (Doubletree Guest Suites) **NEW DATE!** **APRIL 19**  
**SAN FRANCISCO, CA** (Marriott San Francisco) **MAY 13**

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE \$495 REGISTRATION FEE.

Register at [www.sys-con.com](http://www.sys-con.com) or Call 201 802-3069



## Going IDE-less

It was interesting to read in Alan Williamson's editorial "Java 1.4 and the Rest" (Vol. 7, issue 3) that he has foregone the IDE and is going it alone with the JDK. My partner and I started out with JBuilder, and after a couple of years decided to junk it and use UltraEdit and the JDKs. We reasoned that JBuilder was keeping us away from the language and was a barrier to using third-party code and tutorials. We've been really happy with the results (it's been a year) and feel that we have grown in the language enormously. I get a lot of strange looks (especially from software managers who see IDEs as a magic bullet for the "fear of programming" problem) when I tell people that I don't use an IDE. It's nice to hear that others also aren't using it.

Frank Cooley  
ftsoft@computer.org

This is a great editorial. I can relate directly to what Alan Williamson is talking about regarding quality Java architects and also VisualAge.

The blind leading the blind...also the blind teaching the blind (offshore tech schools). And I don't even want to start on my dislike for VisualAge...I would never finish this e-mail.

Although I do enjoy using IDEs such as JBuilder - features such as having a dropdown method list when you type a period after an object are great. I have to admit, there are some definite advantages over the "notepad" IDE.

I've been contracting in the industry for some time and have seen various hiring tactics. For example, one company searching for a lead architect set their entire focus on finding someone who was a guru in VisualAge, WebSphere, and Unix while placing minimal (close to zero) focus on actual architecture experience and techniques. That's like giving someone a job to write articles for a periodical based solely on the fact that that person knows the "ins and the outs" of Microsoft Word.

DJ  
djtatar@hotmail.com

## Assembly Line Development

I think the answer Keith Brown is looking for in his editorial "The IDEal Way Forward for Software Development" (Vol. 7, issue 3) is nothing new at all, it happens to every profession. Programming is going through the same transition that other professions have gone through in the past. Craftsman are only a temporary phenomenon in any profession. How often do you visit your local blacksmith when you need a new part for your car or visit the carpenter to turn a new leg for a broken chair. Blacksmiths and carpenters don't exist in this capacity anymore. Cars are stamped out by other machines and a whole new chair, assembly-line produced, is cheaper than the cost of fixing your broken one.

Software development is making the same transition. Craftsman are simply too expensive; society can't afford to have them building everyday items. As soon as a profession is well-enough understood, engineers will find a way for a machine to

replace the craftsman. "Hard-core" developers do exist: they're building specialized software. But the bulk of the programmers today are piecing together COTS software and painting screens with IDEs. It's simply the profession transitioning from craftsmanship to assembly line.

Scott Wills  
scottw6293@aol.com

## Preferences API Unfairly Categorized

Richard Deadman's article "An API Developer's Primer" (Vol. 7, issue 3) contains many good points, but I believe the author let his biases get the better of him on one point. The article purports to divide APIs into three categories: "dead," "wounded," and "victors."

For the most part his taxonomy is fair: he picked APIs that have been around for a few years and categorized



them based on the success they've achieved.

There is one glaring exception. In the "wounded" category, he names the Preferences API. But this API is brand new! It did not become final until February 2002, as part of the Java 2 Standard Edition Release 1.4. It is simply unknown whether this API will be a dismal failure, a brilliant success, or something in between. In categorizing it as wounded, Deadman may be stating his hopes or his predictions, but he is certainly not stating the facts.

Deadman is entitled to his opinion, but it is worth noting that other commentators don't share it. For example, David Flanagan lists this API as one of the "Top Ten Cool New Features of Java 2SE 1.4" ([www.onjava.com/pub/a/onjava/2002/03/06/topten.html](http://www.onjava.com/pub/a/onjava/2002/03/06/topten.html)).

Time will tell whether the Preferences API succeeds or fails, but responsible journalism dictates that its failure not be announced prematurely.



Josh Bloch  
joshua.bloch@sun.com

*Josh, first of all let me thank you for your e-mail. You're a legend in Java circles, and before now, we haven't had the opportunity to cross paths. It's a shame the first time is a complaint about one of our articles. Richard Deadman was indeed expressing his own views. As it was the sort of piece that had to*

*do this, we allowed him some breadth. Although it is fair to note that he did categorize the API as "wounded" as opposed to "dead," which means he felt it lacking in some respects, thus giving it one down from the "victor" status.*

Alan Williamson



# **SYS-CON**

# **Subscription Offer**

[www.sys-con.com/suboffer.cfm](http://www.sys-con.com/suboffer.cfm)

# **WebLogic**

# **Developer's**

# **Journal**

[www.sys-con.com/weblogic](http://www.sys-con.com/weblogic)

▶ **SavaJe Attains J2SE Compliance for SavaJe OS**  
(Chelmsford, MA) – SavaJe Technologies has announced that it is the first Java licensee to provide a commercially available J2SE-certified operating system for handheld and wireless devices, having passed all the official Sun Compatibility Test Suite (CTS) tests required for J2SE compliance.  
[www.savaje.com](http://www.savaje.com)

▶ **ILOG Releases JRules 4.0**  
(Mountain View, CA) – ILOG, a supplier of software components, has announced the immediate availability of ILOG JRules 4.0, an enterprise-class business rule engine that brings customizable and comprehensive business rule management to enterprises.

Features include the JRules Business Rule Repository, a centralized location for storing business rules and all related information, and event management features that allow users to create rules for detecting complex time-oriented patterns within event flows.  
[www.ilog.com](http://www.ilog.com)

▶ **Borland Debuts Optimizeit Suite and JDataStore 5**  
(San Francisco) – Borland Software Corporation announced the expansion of the Borland Software Platform for Java with the debut of Borland Optimizeit Suite and JDataStore 5 at the JavaOne Conference.

Optimizeit Suite is a performance assurance solution for robust application development that integrates with development and deployment environments.  
[www.borland.com](http://www.borland.com)

## RATIONAL'S NEW ADAPTIVE TEST PLAYBACK TECHNOLOGY

(San Francisco) – Rational's new Adaptive Test Playback technology will provide the industry with self-correcting object recognition, reducing the need to update existing tests throughout the application's life cycle.

Rational's Adaptive Test Playback technology can automatically find target objects without the need for human intervention and time-consuming object remapping.  
[www.rational.com/testjava](http://www.rational.com/testjava)



## JDJ'S APRIL EDITORIAL RALLIES JAVA CAMP

(Montvale, NJ) – SYS-CON Media has published a special editorial, written by JDJ's editor-in-chief Alan Williamson, on its **Java Developer's Journal** Web site. It focuses on the .NET versus J2EE question surrounding the *i*-technology marketplace. The editorial entitled "There May Be Trouble Ahead..." sheds light on the hottest discussion topic of the year ([www.sys-con.com/java/article.cfm?id=1401](http://www.sys-con.com/java/article.cfm?id=1401)).

This editorial was read by more than 100,000 readers within two hours after it appeared on SYS-CON's Web site, and over 500 readers responded within the first two hours of its publication. It was also instantly picked up and simultaneously published on the Slashdot.org and JavaLobby.org Web sites.  
[www.sys-con.com](http://www.sys-con.com)

▶ **Oracle Combines Java-Enabled Mobile Devices and the Enterprise**  
(San Francisco) – Oracle Corp. has announced the immediate availability of the Oracle9i JDeveloper J2ME Add-In and the Oracle9i Application Server (Oracle9iAS) J2ME Developer Kit. Java developers can build and deploy mobile enterprise applications that take advantage of the J2ME standard. Both development tools are currently available for a free download from Oracle's online developer community, Oracle Technology Network (OTN).  
[www.oracle.com](http://www.oracle.com)

▶ **TogetherSoft Delivers Together ControlCenter 6.0**  
(Raleigh, NC) – TogetherSoft Corporation has announced the availability of Together ControlCenter 6.0 for application development. Key enhancements include a new UI builder, improved text editor, new refactorings, and full J2EE 1.3 certification including support for EJB 2.0, Web services support, and a new XML editor.  
[www.togethersoft.com](http://www.togethersoft.com)

▶ **LOOX Releases JLOOXLayout 3.0**  
(San Francisco) – LOOX Software, a subsidiary of eGENUITY Technologies, has released a new version of its Java-based

visualization toolkit. JLOOXLayout 3.0 includes three new layout algorithms for displaying data with pre-existing classes in addition to the Layout Editor

and Layout Manager.  
[www.loox.com](http://www.loox.com)  
[www.engenuitytech.com](http://www.engenuitytech.com)

▶ **Sitraka JClass DesktopViews 6.0 and ServerViews 2.1**  
(San Francisco) – Sitraka has announced the release of Sitraka JClass DesktopViews 6.0, formerly known as JClass Enterprise Suite, now supporting Java 2 SDK version 1.4. JClass DesktopViews provides a range of GUI functionality including 2D and 3D charting, layout and reporting, GUI enhancements, JAR optimization, hierarchical data display, tables and grids, data connectivity, and data input and validation. In addition, JClass ServerViews 2.1 has a set of 100% Java server-side components and includes JClass ServerChart 2.1 and JClass ServerReport 2.1.  
[www.sitraka.com](http://www.sitraka.com)

▶ **JTurbo 3.0 Released**  
(San Francisco) – New Atlanta Communications has announced the availability of JTurbo 3.0, the newest release of its Type 4 JDBC Driver for Microsoft SQL Server, which implements the JDBC 3.0 specification.

Among the added or improved features for JTurbo 3.0 are savepoint support, statement pooling support, improved connection pool configuration, retrieval of parameter metadata, and retrieval of autogenerated keys.  
[www.newatlanta.com](http://www.newatlanta.com)



# EnginData Research

[www.engindata.com](http://www.engindata.com)

# WebSphere Developer's Journal

[www.webspheredevelopersjournal.com](http://www.webspheredevelopersjournal.com)



### DOT.COM

- buyer's guide
- java forums
- mailing list
- java jobs
- java store

### MAGAZINE

- advertise
- authors
- customer service
- editorial board
- subscribe

### CONTENT

- archives
- digital edition
- editorial
- features
- interviews
- product reviews
- source code



**plugged in**

THE JAVA DEVELOPER CONNECTION™ PROGRAM

Get pre-release Java™ technology downloads

REGISTER TO GET THEM

 **Sun**

JAVA take it to the next

## What's Online...

May 2002

### JavaDevelopersJournal.com

Visit [www.javadevelopersjournal.com](http://www.javadevelopersjournal.com) and learn about the latest news and events from the world's leading Java resource. Know what's happening in the industry, minute by minute, and stay ahead of the competition.



### SYS-CON Radio at JavaOne

Listen to host Alan Williamson and other **JDJ** editors as they interview the industry's CEOs, CTOs, and other top management. Tune to [www.sys-con.com/java/javaone2002.cfm](http://www.sys-con.com/java/javaone2002.cfm) and hear discussions on technology, programming, standards, and more with the Java world's top minds.



### JDJEdge 2002 Conference and Expo Register Now for the World's Largest Independent Java Event!

JDJEdge 2002, SYS-CON's third annual Java-focused show, will be held June 24-27 at the Jacob Javits Convention Center along with Web Services Edge 2002 and XMLEdge 2002. The shows will be collocated with TechXNY/PC Expo.



Participate with fellow developers, architects, IT management, and other i-technology professionals in Java Technology Fast Path Certification classes, panel discussions, and three full-day Java classes. Topics include "Java 1.4: What's New," "Building Truly Portable J2EE Applications," "Java Tools for Extreme Programming," and ".NET vs J2EE."

Register online at [www.sys-con.com/jdjudge2002east/registernew.cfm](http://www.sys-con.com/jdjudge2002east/registernew.cfm)

### 2002 Readers' Choice Awards

Make your opinion count. Vote today for the **Java Developer's Journal Readers' Choice Awards**, which have earned the reputation as the most respected industry awards program of its kind. Known as the "Oscars of the Software Industry," this year's awards feature an expanded list of product categories and other additions that will make the fifth annual Readers' Choice Awards the best so far!



### Call for Papers

SYS-CON Events, Inc., invites proposals for speaker consideration at Web Services Edge 2002 West, October 1-3, 2002, at the San Jose Convention Center in San Jose, CA. Ideal candidates are Java developers, Web Services developers, XML developers, .NET developers, industry analysts, IT/IS management, and other industry experts who have first-hand experience in the development or implementation of Web services, Java applications, or XML-based technologies. If you have something substantive, challenging, and original to offer, we encourage you to submit a proposal, and become part of the conference faculty for the i-technology event of the year! Deadline is May 15, 2002. ●




WAKE SOFT  
Learn more now!



Click for a FREE 30-day trial  
AltoWeb



macromedia JRUN



sitiraka



Get SonicMQ™  
sonic SOFTWARE



HERANT DataDirect



Get a Free Java™ Developers Solution CD, Rational XDE™  
Rational



CLICK HERE NOW  
SilverStream eXtend Web Services



spiritsoft  
going beyond jms



Install Anywhere PowerUpdate ZERO G  
www.ZEROG.COM  
Click here to try

SHOP ONLINE AT **JDJSTORE.COM** FOR BEST PRICES OR CALL YOUR ORDER IN AT **1-888-303-JAVA**

BUY THOUSANDS  
OF PRODUCTS AT  
GUARANTEED  
LOWEST PRICES!

GUARANTEED BEST PRICES  
FOR ALL YOUR  
**WEB SERVICES**  
SOFTWARE NEEDS



**MICROSOFT**

\$2,238.99 **Visual Studio .NET Enterprise Architect**

Visual Studio .NET provides developers with the most productive tool for building next-generation applications for Microsoft Windows® and the Web. Visual Studio .NET Enterprise Architect (VSEA) builds on the power of Visual Studio .NET Enterprise Developer by including additional capabilities for designing, specifying, and communicating application architecture and functionality. It enables software architects and senior developers to provide architectural guidance and share best practices across the development team.



**MAINSTAY**

\$124.00 **JustEdit Plus 1.0**

JustEdit Plus is a Java applet offering manual and automatic editing of Web pages anytime, anywhere, using only a Web browser. Manual editing of any target Web page is done in the applet's HTML editing window.



**MICROSOFT**

\$1,629.99 **Visual Studio .NET Enterprise Developer**

With Visual Studio .NET Enterprise Developer, developers can securely version and share their source code, share best practices, target scalable .NET Enterprise Servers, choose from a wide range of third-party tools and technologies, and easily tune the performance of their Web applications and Web Services through the extensive performance-testing tools in Visual Studio .NET.



**WHIZLABS**

\$39.95 **WebSphere@Whiz Certification Simulator**

3 Mock Tests (159 Questions). It comes with a test engine and a question bank of 159 questions on the latest pattern of the IBM WebSphere Certification Exam. It also consists of a diagnostic test which will help you know your strengths and weaknesses so you can plan your preparation accordingly.



**N-ARY**

\$145.00 **n-ary Ticket System**

The Ticket System is a Web-based tool that enables you to log & track important information. The system is extremely flexible and simple to use and can be utilized in numerous different situations. It is a completely 'hands off' system. This means that you do not have to waste time and resources by continuously checking a Web page for updates.



**SILVERSTREAM**

\$495.00 **Extend Application Server Developer Edition (5 User)**

SilverStream eXtend is the first comprehensive, real-world development environment for creating Web Services and J2EE applications. The seamless integration of our proven eBusiness engines and designers gives you the benefits of XML-based, enterprise-wide integration and the power to create, assemble and deploy service-oriented applications.



W W W . J D J S T O R E . C O M

OFFERS SUBJECT TO CHANGE WITHOUT NOTICE

Once you're in it...

- Wireless Business & Technology
- Java Developer's Journal
- XML-Journal
- ColdFusion Developer's Journal
- PowerBuilder Developer's Journal

reprint it...



Contact Carrie Gebert  
201 802-3026  
carrieg@sys-con.com



RePrints

SAVE 16% OFF

**COLD FUSION** Developer's Journal

12 Issues for  
**\$89<sup>99</sup>**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

- Exclusive feature articles
- Interviews with the hottest names in ColdFusion
- Latest CFDJ product reviews
- Code examples you can use in your applications
- CFDJ tips and techniques

That's a savings of **\$17<sup>89</sup>** off the annual newsstand rate. Visit our site at [www.sys-con.com/coldfusion/](http://www.sys-con.com/coldfusion/) or call **1-800-513-7111** and subscribe today!

SAVE 17% OFF

**PowerBuilder** DEVELOPER'S Journal

12 Issues for **\$149<sup>99</sup>**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

- New PowerBuilder features
- Tips, tricks, and techniques in server-side programming
- DB programming techniques
- Tips on creating live PowerBuilder sites
- Product reviews
- Display ads for the best add-on products

That's a savings of **\$31** off the annual newsstand rate. Visit our site at [www.sys-con.com/pbdj/](http://www.sys-con.com/pbdj/) or call **1-800-513-7111** and subscribe today!

# Women's Work

## Competing in the high-tech world



WRITTEN BY  
BILL BALOGLU &  
BILLY PALMIERI

We spoke to three female engineers with 35 years of combined experience working in the trenches of high tech. Each woman has had different experiences and faced numerous challenges. But they all agree that the key difference between male and female engineers is not what they do, but how they do it.

Patty W.'s 18 years of industry experience includes writing commodity trading software and Java Internet security, and working with operating system internals. In her experience, the industry is competitive for men, but for women it's even harder. However, she believes men's culture and upbringing prepares them for it.

Patty has been the sole woman in many groups. "I've felt like I had to work harder to get the job," she says. "And work five times as hard to get recognized. On one contract, a man with 10 years less experience was given the more challenging piece of the job."

Although she's been the senior engineer in many groups, Patty notes that male engineers are less likely to come to her with questions. "Even when there's another woman in the group, she tends to compete with me because she thinks she's being compared with the other woman."

Since 1995 Grace's positions have progressed from Web application engineer to senior engineer to engineering manager. A year and a half ago she made the transition to technical marketing manager, developing demo applications and giving product presentations.

Grace cites the different communication styles of men and women as a key challenge. "Women tend to be consen-

**T**his month we focus on women in the engineering world. In this competitive, male-dominated world, female engineers face a variety of challenges. Many of these challenges have less to do with getting the job done than with getting hired, communicating with their male counterparts, and getting recognized for the work they do.

sus builders, wanting the team to get along well," she says. "The typical male engineer wants to win the argument. In design meetings, whoever can yell the loudest or argue their point best usually wins."

"In that environment you can't try to build consensus, so I've had to become more aggressive in arguing my points," she says, noting that less outgoing male engineers also have difficulty prevailing in such meetings.

Grace traces the lack of female engineers to the way that math and science classes are traditionally taught. "In high school and college almost all my math and science teachers were male," she says. "They taught courses in a direct, fact-based way."

"Men are more likely to memorize facts; women focus more on how will this work in the real world, for the greater good," she says. "I was a tomboy, which helped me in class and to be more competitive in jobs."

Sufie S. considers herself fortunate that she hasn't faced many obstacles as a woman in a man's world. Her BS and MS degrees in computer science and a background in object-oriented design have helped her to move from developer to senior software engineer in her 10-year career.

Balancing her personal and professional lives is a challenge for this working mother of a four-year-old daughter. She'd like to see an engineer's performance measured less by the number of hours worked than by total productivity.

"In the start-up world environment, people who come in late and work late are perceived as working harder than

someone who comes in early and leaves early," she says.

"Good engineers are somewhat punished in this industry," she says. "When you get your work done on time with no problems, you're less recognized than the person who comes in and fixes the bugs at the last minute. That's who becomes the hero of the project."

Sufie would like to see more companies offer telecommuting opportunities, which could help both male and female engineers strike a healthier balance between their personal and professional lives.

"As a contractor I've done two very successful projects from home," she says. "But most companies don't have formal telecommuting policies, so it's hard to get approval for it."

Even if companies make such policy changes, they may come too late for many women in the industry. "I've been hoping for a change since I've been in the industry," says Patty, who reads daily messages on the women in technology, Systems list serv ([www.systems.org](http://www.systems.org)).

"A lot of women are jumping ship because it's so hard to get the respect they deserve," she observes. "Many of them are going into teaching or non-profit work. These jobs don't pay as well, but they're not as competitive."

Although she's not optimistic about the current state of affairs for women in high tech, Patty does have a vision for a brighter future. "If women could get together and start their own company with a more nurturing environment, you could have a very productive company." ●

### AUTHOR BIOS

Bill Baloglu is a principal at ObjectFocus ([www.ObjectFocus.com](http://www.ObjectFocus.com)), a Java staffing firm in Silicon Valley. Bill has extensive OO experience and has held software development and senior technical management positions at several Silicon Valley firms.

Billy Palmieri is a seasoned staffing industry executive and a principal at ObjectFocus. His prior position was at Renaissance Worldwide, where he held several senior management positions in the firm's Silicon Valley operations.

jdcolumn@objectfocus.com



ADVERTISER	URL	PHONE	PAGE
Actuale Corporation	www.actuale.com/info/fbjad.asp	800-884-8665	55
Addison-Wesley			37
Altova	www.altova.com		31
AltoWeb	www.altoweb.com		35
AppDev Training Company	www.appdev.com/promo/MG00052	800-578-2062	72
Apple Computer, Inc.	www.apple.com/macosex	1-800-MY-APPLE	4-5
Ashnasoft Corporation	www.ashnasoft.com		71
BEA	www.bea.com/download		6
Borland Software Corp.	www.borland.com/new/optimizeit/94000.html		79
Canoo Engineering AG	www.canoo.com/ulc/	41 61 228 94 44	25
Capella University	www.capellauniversity.edu	1-888-CAPELLA	34
Compuware Corp.	www.compuware.com/products/optimalj	800-468-6342	21
DataDirect Technologies	www.datadirect-technologies.com	800-876-3101	33
Dice	www.dice.com		41
Dynamic Buyer Incorporated	www.ibm.com/smallbusiness/dynamicbuyer		99
EnginData Research	www.engindata.com		109
ESRI	www.esri.com/mapobjectsjava		39
eXcelon Corporation	www.exln.com	800-962-9620	77
Fiorano Software	www.fiorano.com/tifosi/freedownload.htm	800-663-3621	75
InetSoft Technology Corp.	www.inetsoft.com/jdj	888-216-2353	83
Infragistics, Inc.	www.infragistics.com	800-231-8588	14-15
InstallShield Software Corp.	www.installshield.com	847-240-9111	57
INT, Inc	www.int.com	713-975-7434	32
Interland	www.interland.com	1-866-270-5279	53
ITtoolbox	www.ittoolbox.com		87
JDJ Edge Conference & Expo	www.sys-con.com		91
JDJ Store	www.jdjstore.com	201-802-3012	103
LOOX Software Inc.	www.loox.com	800-684-LOOX	59
Metrowerks Corp.	www.wireless-studio.com		11
Mongoose Technology	www.portalstudio.com		49
Motorola	www.motorola.com/developers/wireless		69
New Atlanta Communications	www.newatlanta.com		73
Northwoods Software Corporation	www.nwoods.com/go/	800-434-9820	64
Oracle Corporation	www.oracle.com/javacode	800-633-1072	19
Oracle Development Tools User Group	www.odtug.com	910-452-7444	89
Parasoft Corporation	www.parasoft.com/jdj5	888-305-0041	51
Pramati Technologies	www.pramati.com	877-PRAMATI	81
Precise Software	www.precise.com/jdj	800-310-4777	23
Prentice Hall PTR			45
QUALCOMM Incorporated	http://brew.qualcomm.com/ZJD4		63
Quintessence Systems Limited	www.in2j.com		61
Rational Software	www.rational.com/offer/javacd2		43
SilverStream Software	www.silverstream.com/coals	1-888-823-9700	29
Sitraka	www.sitraka.com/class/jdj	800-663-4723	17
Sitraka	www.sitraka.com/jprobe/jdj	800-663-4723	67
Sitraka	www.sitraka.com/performance/jdj	800-663-4723	116
Simplex Knowledge Company	www.sk.com	845-620-3700	97
Softwired, Inc.	www.softwired-inc.com	41-14452370	85
Sonic Software	www.sonicsoftware.com	800-989-3773	2
SpiritSoft	www.spiritsoft.com/climber		27
Sprint PCS	http://developer.sprintpcs.com		65
Sun Microsystems	www.sun.com/forte		13
SYS-CON Industry Newsletters	www.sys-con.com	201-802-3020	95
SYS-CON Subscription Offer	www.sys-con.com/suboffer.cfm		107
TogetherSoft Corporation	www.togethersoft.com/1/jdj.jsp	919-833-5550	8
WebGain, Inc.	www.webgain.com/toplink_create3.html	1-877-WebGain Ext.15858	115
WebLogic Developer's Journal	www.sys-con.com/weblogic	800-513-7111	107
Web Services Edge Conference & Expo	www.sys-con.com		93
Web Services Edge World Tour 2002	www.sys-con.com	201-802-3069	104-105
Web Services Journal	www.sys-con.com	800-513-7111	101
WebSphere Developer's Journal	www.webspheredevjournal.com	800-513-7111	109
Zero G	www.zerog.com	415-512-7771	3

**General Conditions:** The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *Java Developer's Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in *Java Developer's Journal*. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.

# Next Month

## Document Printing in Java

Keith G. Gauthier shares his simplified process for printing Java documents.

## Interfacing to AIM with Java

Jeff Heaton shows how to create a reusable class that allows access to America Online's (AOL) Instant Messenger network.

## Hello World! in 71 Bytes

Norman Richards explains how he used Java class files and the Java Virtual Machine to create a small Java "Hello World!" program.

## Using the Java Native Interface Productively

Andrew J. Chalk addresses the problem of using the JNI in a production situation and presents a set of techniques that simplify most of the repetitive tasks of passing information to the native code and getting it back into your Java program.

## Broken Windows in the Java World

Small disorders can lead to larger ones, and perhaps even to project cancellations and firings. Joe Xu shares a practical approach to identifying and fixing broken Java windows.

# Fear of the Dark



WRITTEN BY  
BLAIR WYMAN

**A**s I'm pathologically fond of pointing out – I'm a child of a bygone era. Oh, I'm not old enough to remember the time before Sputnik, light bulbs, or the coagulation of the planets from protostellar dust clouds, but I surely do remember GI Joe at 45 caliber, Ray Stevens at 45 RPM, and factory-rolled cigarettes at 45¢ a pack sold to anxious minors desperate to rebel (just like everyone else).

It seems ironic to me that this column – gilding the edge of *JDJ's* ultramodern fabric – deals mostly with dusty memories of my vanished youth. On the other hand, I've heard that I should "write about what I know," so my options are extremely limited. Extremely. I mean, how many cab-driving stories can a person take? (Many would say that one is already too many.)

I suppose I could repeatedly spew the first 50 digits of pi – which I happened to memorize as a teenaged proto-geek – but that would be almost as stupid as the original act of memorizing it. I've always thought that those brain cells I still use to hold pi might have otherwise created some history-making invention, or developed the cure for the common cold, or at least been happily sacrificed on the altar of top-shelf single-malt Scotch whiskey. Never too late to hope, though, I guess.

Uh-oh, here it comes, I feel a recitation coming on:

"3.1415926535897932384626433832795028  
8419716939937510"

< sigh > Boring, eh?

## AUTHOR BIO

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

My sincerest hope, dear reader, is that perhaps I can finally count my own self among the "things I know," and that you will enjoy some of these vacuous recollections from the goofy amalgamation of quarks that is my personal "main storage."

And, by all means, if I'm so utterly irrelevant or patently offensive that you're thinking about dropping your *JDJ* subscription, please drop me a line so I can quit this gig.

So, to the task at hand: this month's "stuff."

As I think back – searching for this month's **Thread** fodder in that blurring database of murky self-awareness – certain recollections stand out boldly in my memory, maintained with (at least the illusion of) crystalline clarity. Some such moments are undoubtedly the result of a simultaneous psychic shock – the assassinations of JFK, MLK, RFK, and John Lennon; the first moon landing; the Challenger disaster; 9/11.... Others are seemingly just random snippets of remembered awareness, inexplicably preserved for personal perusal.

For example, something just reminded me of one particular evening in December of 1967. I was 10 years old and had come home from a friend's house in time to see the name of Dr. Christiaan Barnard as it flashed on the TV screen. It seemed to me that "Dr. Christiaan Barnard" had a pretty dog-goned funny way of spelling his first name, but the venerable Mr. Cronkite didn't even crack a smile as he told us the incredible news: this remarkable man from South Africa had successfully performed the first human heart transplant.

Twilight was just giving way to evening as Walter awed us all with this landmark news. What promise the world

held! What wonderful cause for hope and optimism and pride in the human race! We cured polio, managed to forestall nuclear holocaust (so far), and now were on the verge of removing and replacing the human heart.

However, lovely evenings during a South Dakota winter don't usually last very long, and this was no exception.

Through the darkly colored glass of distant recollection, it seems as though darkness fell too swiftly, as Mr. Cronkite inevitably turned to the daily body counts out of Vietnam: so many dead, so many wounded, so many missing in action. So very, very many. Reality blends smoothly into surreality, as I remember the shards of shattered optimism quietly exploding at the sight of graphic footage. Film at 11.

While I can nearly picture Walter's face in my memory, I can't remember the actual body count from that particular day. One thing is certain, though, there was a body count that day; there was a body count every day.

I sure hope I'm wrong, but it looks to me as if the world may be in for another extended bout of nightly body counts on the "Nightly News." Is there anything we can do to stop it? Should it be stopped? Don't ask me. I'm just a family man, computer programmer, and ersatz freelance writer. ☘

blair@blairwyman.com



# WebGain, Inc.

[www.webgain.com/toplink\\_create3.html](http://www.webgain.com/toplink_create3.html)



# Sitraka

[www.sitraka.com/performasure/jdj](http://www.sitraka.com/performasure/jdj)