

JAVA DEVELOPER'S JOURNAL

# JDJ

FEBRUARY 2004 VOLUME:9 ISSUE:2



**BENFIELD ON  
i-TECHNOLOGY**

...FROM THE FOUNDING EDITOR

**PLUS\_**

**Mobility:** Go Wild Wirelessly  
with **BLUETOOTH**

**Color Management:** Using **COLOR**  
Technology in Java

**Java Enterprise Viewpoint:** Our **ROOTS**

**EXCLUSIVE...**

# THE END

— OF —

**MIDDLEWARE**  
by Sun's **Jonathan Schwartz**

RETAILERS PLEASE DISPLAY  
UNTIL APRIL 30, 2004

\$5.99US \$6.99CAN



# BEA WebLogic Workshop 8.1 makes you more

# AMAZING.



Development Tools  
December 30, 2003  
BEA WebLogic Workshop 8.1  
BEA Systems, Inc.

"A shared, consistent  
development platform...for  
J2EE experts and business-  
oriented developers alike."



Java Pro Readers  
Choice Award

"WebLogic Workshop  
empowers all application  
developers...not just J2EE  
developers."



Software Development Jolt  
Product Excellence Award

"The integration of an IDE,  
controls and a deployment  
environment...greatly enhances  
developer productivity."



CRN Test Center  
Product of the Year

"By far the most innovative  
development tool reviewed  
this year...when compared  
with other tools, Workshop  
blew away the competition."

But don't take their word for it, find out for yourself now, for free.

BEA's WebLogic Workshop 8.1 has won every major award for software development tools in the past year. It's a full-featured Java development environment that lets you visually build and assemble enterprise-scale Web Services, Web Applications, JSPs, Portals, EJBs, and Business Process models based on the latest standards and open source technologies.

Get it now. For free. Visit <http://dev2dev.bea.com/workshop1>

**dev2dev**<sup>TM</sup>

**bea**<sup>®</sup>



## How Can You Sink Your Application Overhead?

With Cyanea/One®, your company can ensure that the resource consumption of its J2EE applications stays below desired thresholds. Through Cyanea/One's Performance Analysis and Reporting (PAR) engine, you can quickly baseline an application and trend its performance and resource utilization over time. Automatically identify hotspots within the code and take corrective action. Be instantly notified when key threshold conditions are approaching. Improve service levels and customer satisfaction. Accomplish all these without touching your code or even requiring knowledge of your application.

*...Keeping* **Your Applications Under PAR.**



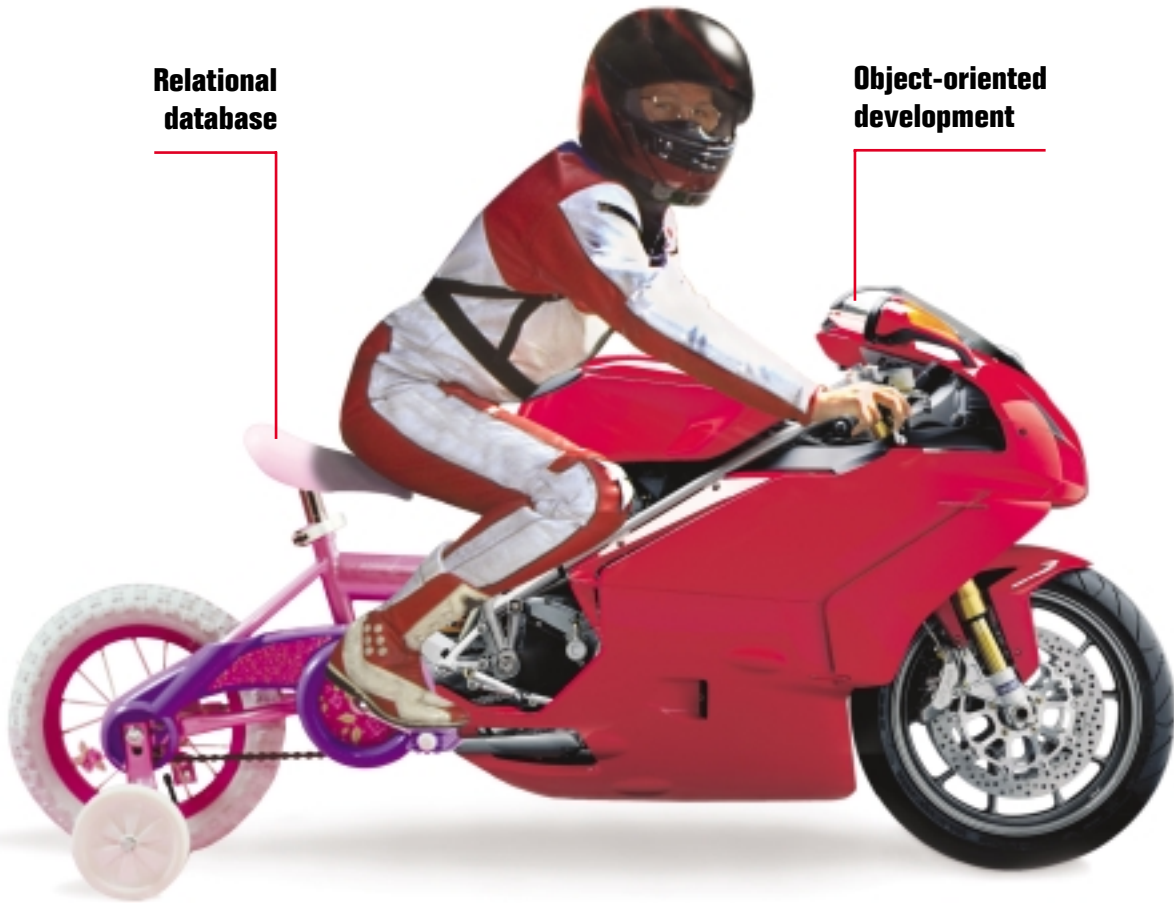
Find PAR at:  
[www.cyanea.com/wsdj/underpar.html](http://www.cyanea.com/wsdj/underpar.html)  
1-877-CYANEAS | [sales@cyanea.com](mailto:sales@cyanea.com)

Cyanea® and Cyanea/One® are registered trademarks of Cyanea Systems Corp.



**Relational  
database**

**Object-oriented  
development**



# A BETTER DATABASE CAN SPEED UP YOUR DEVELOPMENT CYCLE

If your back-end database isn't a good match for your front-end development, you need a new database.

Caché is the *post-relational* database that combines high-performance SQL for faster queries and an advanced object database for rapidly storing and accessing objects. With Caché, no mapping is required between object and relational views of data. That means huge savings in both development and processing time.

Applications built on Caché are massively scalable and lightning-fast. Plus, they require minimal or no database administration.

More than just a database system, Caché incorporates a powerful Web application development environment

that dramatically reduces the time to build and modify applications.

The reliability of Caché is proven every day in “life-or-death” applications at thousands of the world’s largest hospitals. Caché is so reliable, it’s the leading database in healthcare – and it powers enterprise applications in financial services, government and many other sectors.

We are InterSystems, a specialist in data management technology for twenty-five years. We provide 24x7 support to four million users in 88 countries. Caché is available for Windows, OpenVMS, Linux and major UNIX platforms – and it is deployed on systems ranging from two to over 10,000 simultaneous users.



**Try a better database. For free.**

Download a free, fully-functional, non-expiring version of Caché or request it on CD at [www.InterSystems.com/match3](http://www.InterSystems.com/match3)



# JDJ contents

Cover Story: **Industry Perspective**

*JDJ* Exclusive...

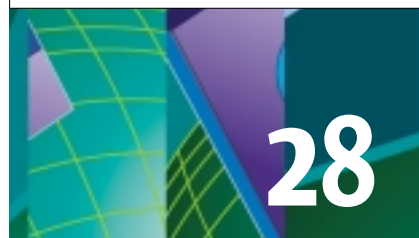
## Jonathan Schwartz

*EVP of Software, Sun Microsystems*

16



**Features**



28

**The Delegation-Managed Persistence Entity Bean**

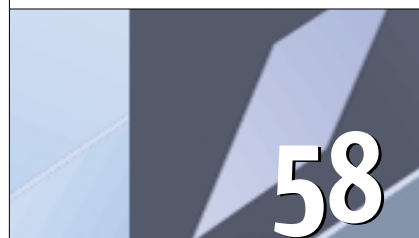
by Tal Cohen



42

**Java New Input/Output**

by Vishwanath K



58

**No Fluff, Just Stuff:  
An Interview with Jay Zimmerman and Ted Neward**

by Kirk Pepperdine

FROM THE FOUNDING EDITOR  
**Development Tools for All**  
by Steve Benfield .....6

FROM THE EDITOR  
**Keeping the Faith**  
by Joe Ottinger .....8

JAVA ENTERPRISE VIEWPOINT  
**Our Roots**  
by Kirk Pepperdine .....10

USE CASES  
**HTTP Session Garbage Collector**  
*Removing cached data*  
by Abhinasha Karana .....12

MOBILITY  
**Go Wild Wirelessly with Bluetooth and Java**  
*Developing portable Bluetooth apps*  
by Ben Hui .....20

CORE AND INTERNALS VIEWPOINT  
**Man with an Open Heart**  
by Jason Bell .....32

FRAMEWORKS  
**Java Collections**  
*Managing collections*  
by David McReynolds .....34

LABS  
**DevPartner 3.0.1 Java Edition**  
*by Compuware Corporation*  
Reviewed by Vijay Phagura .....50

DESKTOP JAVA VIEWPOINT  
**Behind the Glass**  
by Joe Winchester .....52

GUI  
**Turning Components into Domain GUI Objects (DGO)**  
*Improving your design*  
by Ted M. Young .....54

COLOR MANAGEMENT  
**Using Color Technology in Java**  
*Heighten the visual impact of your application*  
by John Chamberlain .....56

LABS  
**Droplets by Droplets Inc.**  
Reviewed by Somnath Banerjee .....62

JSR WATCH  
**From Within the Java Community Process Program**  
*From proposals to final approvals*  
by Onno Kluyt .....64

@ THE BACKPAGE  
**Offshore Outsourcing: Magic Bullet or Dirty Word?**  
*It all depends on your perspective*  
by Jack Martin .....66

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for \$69.99 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: JDJ, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.



Steve Benfield

# Development Tools for All

**M**y dad is a DBA. However, he doesn't design large databases, he doesn't write extremely elaborate multiselect SQLs (I don't think he's ever read a Joe Celko book), and he certainly doesn't care about the latest, greatest news in the world of technology. He's been at the same place for about 15 years, is respected by his co-workers, and makes sure that the rest of his organization gets the information necessary to get their jobs done.

I have another friend who is a programmer. She works with Visual Studio, cranks out form-based apps that attach to data, and basically helps build client/server apps for her organization. The apps she builds are used by lots of people in her organization, thereby she provides value and remains employed.

Neither one of them really cares about the latest battles over Web services standards, about the Microsoft versus Linux/open source wars, about the JCP process, or desktop Linux. They represent a large percentage of the people you meet in typical IT organizations; they make up the teams that develop applications that you use in your organizations. I call them "business developers."

The other primary group in IT development organizations is the "technology elites." These are the architects and senior developers who know how to use UML correctly, who know the latest Java standards, and who can write complex server code with ease. Tools used by this group need to include analysis/design tools, strict standards compliance, automation/generation of redundant things, and most of all – the ability to circumvent the tool and get straight to the code without interference.

Typically this group of technology elites might represent 10% of an IT development organization. They tend to be the ones who make decisions on which tools and technologies to use and drive the technical direction of the company. The fact that you're reading this magazine means there's a high probability that you're a technology elite.

Business developers can create GUIs, write HTML and JavaScript, build workflows, execute services, make maintenance changes, etc. Given a decent set of specs, some existing services to call, and a productive GUI that abstracts the low level, this group can produce lots of appli-

cations and contribute huge value to the IT organization. They build the day-in and day-out applications that your employees, customers, and partners run. These developers don't really care if what they build follows the latest industry standards, uses entity beans, or uses the hottest new products, it's their tool and its productivity that is important to them. They just want to get applications out the door; as long as some of the basics are covered, like running on top of a J2EE or servlet engine, they are happy.

The problem is that creating productivity out of a set of industry standards is complicated. Real productivity and automation of development tasks requires an advanced (aka proprietary) framework and an intuitive GUI. Every vendor that has a solution in the Java J2EZ space (Making J2EE easier= J2EZ) has a proprietary solution they're selling. I'm not characterizing this as a bad thing, just a necessary thing and something you need to realize when you go for productivity.

BEA and IBM have made strides in productivity but their solutions only work on their application servers and platforms, and they still miss the mark for the vast majority of business developers. While they do support standards, they fall short of the J2EE promise of platform independence. Smaller vendors build systems that will work on all the major Java platforms. Borland and Compuware (and IBM Rational) have great tools that target all the platforms, but they're geared toward technology elites and model-driven development. M7 and Exadel have built excellent environments for automating Struts/JSP application development, but there are questions on how usable they are for the business developer. Then you have vendors, such as ClearNova, who have solutions based on J2EE but have decided to go for massive productivity by sacrificing some standards support such as Struts.

There are many approaches to application development, and they have to suit the needs of your development organization. If your group is mostly technology elites, there are plenty of tools for you to look at. If you have a mix, then it might mean you need some tools for both groups. If your team is mostly business developers and speed of development and productivity is the name of the game, then a slightly less-standard approach might be in order. ☺

Steve Benfield is an independent consultant based in Atlanta. Previously he was CTO of SilverStream Software. Steve was the editor of SYS-CON's first magazine, *PBDJ*, in 1994. [steve.benfield@earthlink.net](mailto:steve.benfield@earthlink.net)

**International Advisory Board**

|                 |               |
|-----------------|---------------|
| Calvin Austin   | (Sun)         |
| Jason Bell      | (Independent) |
| Jason Briggs    | (Independent) |
| Jeremy Geelan   | (SYS-CON)     |
| Thorsten Laux   | (Sun)         |
| Rickard Öberg   | (Independent) |
| Joe Ottinger    | (Independent) |
| Bill Roth       | (E.piphany)   |
| Ajit Sagar      | (Independent) |
| Eric Stahl      | (BEA)         |
| Jon Stevens     | (Apache)      |
| Aaron Williams  | (JCP)         |
| Alan Williamson | (SYS-CON)     |
| Joe Winchester  | (IBM)         |
| Blair Wyman     | (IBM)         |

**Editorial**

|                            |                 |
|----------------------------|-----------------|
| Editor-at-Large:           | Alan Williamson |
| Editor-in-Chief:           | Joseph Ottinger |
| Executive Editor:          | Nancy Valentine |
| Java-Enterprise Editor:    | Kirk Pepperdine |
| Core and Internals Editor: | Jason Bell      |
| Desktop Java Editor:       | Joe Winchester  |
| Gaming Editor:             | Jason R. Briggs |
| Contributing Editor:       | Ajit Sagar      |
| Contributing Editor:       | Glen Cordrey    |
| Founding Editor:           | Sean Rhody      |

**Production**

|                         |                     |
|-------------------------|---------------------|
| Production Consultant:  | Jim Morgan          |
| Associate Art Director: | Tami Beatty         |
| Associate Editors:      | Jamie Matusow       |
|                         | Gail Schultz        |
|                         | Jean Cassidy        |
|                         | Jennifer Van Winkel |
| Online Editor:          | Lin Goetz           |
| Technical Editor:       | Bahadir Karuv, PhD  |

**Writers in This Issue**

Somnath Banerjee, Jason Bell, Steve Benfield, John Chamberlain, Tal Cohen, Ben Hui, Jeremy Geelan, Abhinasha Karana, Viswanath Krishnan, Onno Kluyt, Jack Martin, David McReynolds, Joseph Ottinger, Kirk Pepperdine, Vijay Phagura, Joe Winchester, Ted Young

To submit a proposal for an article, go to <http://grids.sys-con.com/proposal>

**Subscriptions**

For subscriptions and requests for bulk orders, please send your letters to Subscription Department [subscribe@sys-con.com](mailto:subscribe@sys-con.com). Cover Price: \$5.99/issue. Domestic: \$69.99/yr. (12 Issues) Canada/Mexico: \$99.99/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: 201 782-9638

Java Developer's Journal (ISSN#1087-6944) is published monthly (12 times a year) for \$69.99 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 © 2004 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 Media and SYS-CON Publications, Inc., reserve the right to revise, republish and authorize its readers to use the articles submitted for publication.

Worldwide Newsstand Distribution  
Curtis Circulation Company, New Milford, NJ  
For List Rental Information:  
Kevin Collopy: 845 731-2684, [kevin.collopy@edithroman.com](mailto:kevin.collopy@edithroman.com)  
Frank Cipolla: 845 731-3832, [frank.cipolla@epostdirect.com](mailto:frank.cipolla@epostdirect.com)

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.





**WILL IT WORK?**  
SHOULD YOU EVEN HAVE TO ASK?

Data connectivity is vitally important to the health of your applications. Is this really where you want to take chances? Your data connectivity choices can have dramatic effects on scalability, interoperability and performance. And you'll be left facing increasing development, maintenance and testing costs plus potential loss of revenue. DataDirect offers the industry's most comprehensive, proven suite of database-independent Type 4 JDBC drivers. Our extensively tested J2EE-certified drivers include the most advanced JDBC 3.0 features including Distributed Transaction Support, Connection Pooling and BLOB/CLOB support. DataDirect Connect<sup>®</sup> for JDBC<sup>®</sup> is the SPECjAppServer and ECperf performance leader.

**DataDirect**<sup>™</sup>  
TECHNOLOGIES

[www.datadirect.com](http://www.datadirect.com)  
800-876-3101

Find out what else you might be missing. Download our whitepaper,  
"What you don't know about database drivers CAN hurt you" @ [www.datadirect.com/JDJ](http://www.datadirect.com/JDJ)





**Joe Ottinger**  
Editor-in-Chief



## Keeping the Faith

In the Java community you have two schools of thought: the zealots, if you will, who feel that pure Java is worth the attempt, and the compromisers, who feel it's more important to use Java no matter what.

Swing against SWT is a good example of this: SWT is a compromise, where native GUI elements are used to further Java, and Swing is the pure Java element. The GUI is hardly the only battle fought by these two "camps." Think of those who advocate Java-to-native compilation (or oppose it), or those who want Java to have features used by C#, like attributes and autoboxing (and those who don't want those features). They're similar issues, fought for and over with the passion normally reserved for raw survival.

I find myself on the zealots' side, but I have to confess that I understand those who do not. Perhaps the zealots – the "pure Java" camp – are throwing the dice, hoping Java is strong enough right now to survive and win. Throwing the dice means you might win...and you might lose – and Java might become irrelevant. Compromising might subvert the original intent, but also ensure the survival of the technology and the benefits it has brought and continues to bring.

It could be that the time is ripening for open source Java, with the commits being validated by Sun to prevent wild strains in a methodology similar to that followed by Linux. Java is currently burdened by its appearance as both a product (see the "Java Desktop," for example) and a commodity (witness the various add-ons, such as SWT and JGoodies, which purport to make "core Java" better or easier or, if you like, "faster"). The add-on products aren't bad, per se,

but with the confusion over Java's core role, they fracture the foundation for the community. We need to see Java as a commodity, as something that everyone can use, such that Sun is itself a controlling interest of an available technology – not the controlling interest of a product that everyone else is allowed to use, because the allowance itself grants value. It's either valuable or it's not. I say it is – but the longer it's shared grudgingly, the less valuable it appears.

What I would like to see is a viable future strategy. I'm willing to accept that SWT fixed some issues present in an older version of the JRE, and the native OS look and feel is an advantage for those who wish it. (I personally switch OSES too often to want a native look and feel. I want my applications to work the same regardless of OS.) That said, I think that Swing has caught up to SWT in many ways and, in other ways, I think it will catch up if it's important enough, and eventually you'll see SWT as I do: as a split in Java, in what could be a unified front. It's important to me that Java has a plan for handling situations like this, where there's a viability in "pure Java" that needs to be preserved, but an "impure Java" possibility needs to be addressed.

I wish I could see the future and tell you which camp was better for Java: the one that advocates a "pure vision," despite being flawed in perceptible ways, or the one that advocates the surrender of a fight that, in their opinion, not only isn't winnable, but has already been lost. The optimist in me says that the former view is better, that flaws can be corrected with time...but the existence of the latter worries me, unless steps are taken to use the strengths of all involved. ☉

Joseph Ottinger is a consultant with Fusion Alliance ([www.fusionalliance.com](http://www.fusionalliance.com)) and is a frequent contributor to open source projects in a number of capacities. Joe is also the acting chairman of the /DJ Editorial Advisory Board.

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



President and CEO:  
**Fuat 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)  
Group Publisher:  
**Jeremy Geelan** [jeremy@sys-con.com](mailto:jeremy@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)  
Director, Sales and Marketing:  
**Megan Ring** [megan@sys-con.com](mailto:megan@sys-con.com)  
Advertising Sales Managers:  
**Alisa Catalano** [alisa@sys-con.com](mailto:alisa@sys-con.com)  
**Carrie Gebert** [carrie@sys-con.com](mailto:carrie@sys-con.com)  
Associate Sales Managers:  
**Kristin Kuhnle** [kristen@sys-con.com](mailto:kristen@sys-con.com)  
**Beth Jones** [beth@sys-con.com](mailto:beth@sys-con.com)

### Editorial

Executive Editor:  
**Nancy Valentine** [nancy@sys-con.com](mailto:nancy@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)  
**Jennifer Van Winkel** [jennifer@sys-con.com](mailto:jennifer@sys-con.com)  
Online Editor:  
**Lin Goetz** [lin@sys-con.com](mailto:lin@sys-con.com)

### Production

Production Consultant:  
**Jim Morgan** [jim@sys-con.com](mailto:jim@sys-con.com)  
Lead Designer:  
**Tami Beatty** [tami@sys-con.com](mailto:tami@sys-con.com)  
Art Director:  
**Alex Botero** [alex@sys-con.com](mailto:alex@sys-con.com)  
Associate Art Directors:  
**Louis F. Cuffari** [louis@sys-con.com](mailto:louis@sys-con.com)  
**Richard Silverberg** [richards@sys-con.com](mailto:richards@sys-con.com)

### Web Services

Vice President, Information Systems:  
**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

Accounts Receivable:  
**Charlotte Lopez** [charlotte@sys-con.com](mailto:charlotte@sys-con.com)  
Financial Analyst:  
**Joan LaRose** [joan@sys-con.com](mailto:joan@sys-con.com)  
Accounts Payable:  
**Betty White** [betty@sys-con.com](mailto:betty@sys-con.com)

### SYS-CON Events

President, SYS-CON Events:  
**Grisha Davida** [grisha@sys-con.com](mailto:grisha@sys-con.com)  
Conference Manager:  
**Lin Goetz** [lin@sys-con.com](mailto:lin@sys-con.com)  
National Sales Manager:  
**Sean Raman** [raman@sys-con.com](mailto:raman@sys-con.com)

### Customer Relations

Circulation Service Coordinators:  
**Shelia Dickerson** [shelia@sys-con.com](mailto:shelia@sys-con.com)  
**Edna Earle Russell** [edna@sys-con.com](mailto:edna@sys-con.com)  
**Linda Lipton** [linda@sys-con.com](mailto:linda@sys-con.com)  
|DJ| Store Manager:  
**Brunilda Staropoli** [brunilda@sys-con.com](mailto:brunilda@sys-con.com)



Leaner, Meaner, Faster Java Development.

**Borland® JBuilder® Developer**, from the #1 Java IDE company in the world. It's all the power you crave. Yet lightweight and agile. At a price that won't leave you grounded. Automate the routine stuff. Handcraft the unique. Blast through every stage of the process, with more bullet-proof results. Whether your app is headed to the desktop, Web, or mobile, Borland JBuilder Developer gets you up and going fast. And lands the product flawlessly.

- Customizable code editor with CodeInsight™ and ErrorInsight™
- Import project source from any IDE or editor
- Two-way visual Struts designer
- JSP™ Tag Library/framework support
- Local and remote servlet/JSP debugging
- XML and database tools
- Develop, debug and deploy mobile applications
- Integrated unit testing
- Advanced build and configuration management with Apache™ Ant
- Archive builder
- OpenTools API

Take a test flight today: Sit down, buckle up and hang on at [go.borland.com/j3](http://go.borland.com/j3)

Made in Borland® Copyright © 2004 Borland Software Corporation. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. • 21431

**Borland®**  
Excellence Endures™



**Kirk Pepperdine**  
Java Enterprise Editor

## Our Roots

**D**o you enjoy history? I do. In fact, I've always enjoyed history for I've always found that understanding the past has been useful in helping me to understand the here and now. Part of my here and now is the taking on of the role of enterprise editor. The question for me is, how did I get here?

When I look back at my career, I realize that my choices have allowed me to be closely aligned with many aspects of core J2EE technologies for more than 15 years. Yes, I do know that J2EE has not been here for 15 years, but when you look past that exact moment in time when the idea for J2EE was first hatched, you can see a vast trail of technology and research that has gotten us to where we are today. My interest in history has led me to follow that trail and, as is the case with most historical tales, I was taken by surprise by just how far back you need to travel. Take the virtual machine, for instance.

for a number of years. The difference is that Java is really the first commercially successful language/platform to support distributed computing. From this success came the realization that to truly bring distributed computing to the masses, they would need help putting it all together.

The help first came in the form of the Enterprise JavaBean application server. I'm unaware of the true origins of the application server but I have worked with one for more than 10 years – GemStone for Smalltalk (GS/S). GS/S is generally recognized as an awesome collection of technologies. It was supported by an orthogonal transactional persistence mechanism that was instrumental in its support of distributed computing. As good as the technology was, people had difficulties understanding it. Though you could think of it as an object database that relied on a proprietary Smalltalk implementation, it was really more than just that. It was just when

“ In all of that confusion, we have been able to define and build a technology that is pretty solid”

In 1965, a team of engineers working for IBM first devised a virtual machine so they could study multiprogramming operating systems. What they meant by virtual machine at that time was a duplicate of the actual machine they were testing on, but the VM had less memory. If you think about the Java Virtual Machine, the definition is just about the same. Okay, there's not a real Java machine per se, but the JVM does have less memory than what is available in the underlying hardware/OS.

From these raw beginnings came Dijkstra's work, where he built an operating system by layering a number of VMs on top of each other. Each VM was an abstraction of some underlying portion of the OS. What followed was yet another improvement by another IBM team, which resulted in the creation of the VM/CMS, a time-sharing OS for the IBM 370. Right around that time, Alan Kay laid the groundwork for Smalltalk, considered to be the de facto standard for OO technologies. This brief paragraph is only the tip of the iceberg of the research and development that really started in the late '50s. Just from this little bit of history, we can see through the marketing hype and understand why Java is not just a language – it truly is a technology platform.

Legend has it that Bill Joy conceived of the principles behind Java in the late 1970s, but it wasn't until the early '90s that the necessary ingredients came together to make it a reality. Even as early on as the late '70s, Smalltalk, Lisp, and a number of obscure languages had already been operational

I started working for GemStone that I made the jump to Java. GemStone had begun the process of porting their Smalltalk technology to create an early version of a Java application server. Again, GemStone was ahead of the curve as they were working without a specification, so they defined a distributed component model based on JavaBeans. Not long after the initial implementations of GS/J went to press, the EJB spec appeared and everyone started the race to implement the latest features in the latest specification. After 15 years in the business, the appearance of the EJB specification provided GemStone with a means to describe their technology.

I began my journey in a world that was as aware of the possibilities of Java as it was unaware of its absence. The journey has been exciting as I've been able to work with a number of world-class engineers who were implementing the latest bleeding-edge technology in a market that was still trying to figure out what it wanted. Yet, in all of that confusion, we have been able to define and build a technology that is pretty solid. It has let us mere mortals build systems that would have been unthinkable only a few years ago. But in this ever-changing game of cat and mouse, the cats (our users) continue to adjust their expectations and demand even more complex systems so we still have plenty of problems to solve. Having said this, maybe there are hints to the solutions to these new challenges somewhere in our collective past. Certainly, we should take the time to look for them. ☺

Kirk Pepperdine is the chief technical officer at Java Performance Tuning.com and has been focused on object technologies and performance tuning for the last 15 years. Kirk is a co-author of *Ant Developer's Handbook* (Sams).

kirk@javaperformancetuning.com





# Optimize J2EE.

The J2EE revolution is here to increase performance, value, and lower IT costs. It's a solution from Mercury Interactive that makes your whole J2EE applications ecosystem work right.

It's the fastest way to find the toughest J2EE problems from the development to the live applications. Even pinpoints root cause and the line of source code.

It's about more than delivering J2EE applications. It's about delivering applications that work and yield real business value.

It's the Business Technology Optimization revolution.

And Mercury is the only one bringing you a revolutionary solution for J2EE.

Download our free white paper, "**Diagnosing J2EE Performance Problems Throughout the Application Lifecycle,**"

at [www.mercuryinteractive.com/optimizej2ee](http://www.mercuryinteractive.com/optimizej2ee)

Get Optimized™



# HTTP Session Garbage Collector

by Abhinasha Karana

## Removing cached data

A common approach to caching data in Web applications is to use an HTTP session. A business use case that spans multiple HTTP requests may create the need for caching in a Web tier. Once business use-case processing is completed, this cached data needs to be removed. Failure to do this may lead to memory leakage, which becomes noticeable when a user HTTP session continues for hours.

### The Solution

The following terminology is used in subsequent sections:

- A use case is a sequence of steps performed by a user toward realization of a business requirement.
- A nested use case is the extension point from the base use case.
- The use case context represents the logical starting point of a use case.
- The handler is a server-side component responsible for processing HTTP requests.
- The cache element is an object, cached during use case processing.
- The navigation path is the sequence of handlers invoked when fulfilling a use case.

Let's begin by considering a few basic rules.

**Rule 1:** *Only one use case is active at any instance: A user cannot process multiple use cases at a given instance.*

**Rule 2:** *A use case processing stage governs the cache element life cycle:*

1. **Start stage:** New use case processing starts on completion or termination of prior use case processing. Cache elements that correspond to a prior use case are removed.
2. **Intermediate stage:** The user navigates to the next step or to a prior step of the use case. Forward navigation adds new cache elements; backward navigation may remove cache elements on an as-needed basis.
3. **End stage:** On completion of the use case, all accumulated cache elements are removed.

To implement this rule we need to provide an additional parameter to identify the use case processing stage.

**Rule 3:** *Handlers operate inside a use case context:* User navigation dynamically builds a cache hierarchy tree with the handler and use case context as nodes and the cache element as a leaf. The handler node could hold a successor handler, cache elements, and a nested use case context. The use case context node holds a start handler. The root of the tree is the use case context.

**Rule 4:** *Handlers operate on cache elements: Handlers create, read, and destroy cache elements.*

Keeping these rules in mind, let's address the stated problem. We have conceptually built a cache hierarchy tree, which is the key concept to addressing the problem.

**Step 1:** *Each request will go through the cache garbage collector*

At the start of a new use case, the cache garbage collector removes all existing cache elements of the previous use case.

Figure 1 addresses the memory leakage problem. The intro-

duction of the cache hierarchy tree removes tight coupling between handlers.

**Step 2:** *Handlers navigate the cache hierarchy tree*

A pointer is maintained to navigate the cache hierarchy tree. It points to the most recently processed node in the use case context. With the help of this pointer we can:

- Jump to a previous step
- Jump the nested use case context

During a jump operation all cache elements that are attached to downstream handlers may be removed. It's helpful in the following business scenarios: backward navigation and use case termination.

Efficient memory handling at a granular level is achieved. At the same time, the decoupled handlers make future enhancements easier.

### Solution Implementation

#### Cache Hierarchy Tree Design

The cache hierarchy tree is built using the GoF Composite pattern (see Figure 2). The use case context and handlers are the nodes; the cache element names are leaves; the actual cache element instances reside in the HttpSession.

#### Cache Hierarchy Tree Management

The "UserAppStateContainer" object holds the cache hierarchy tree. This container instance is stored in the user's HTTP session. "UserAppStateManager" helps manage the cache hierarchy tree.

### Conclusion

The cache hierarchy tree empowers HTTP session garbage collection; however, this tree is also kept in the user's HTTP session. So in a clustered environment it also needs to be replicated. ☺



Abhinasha Karana

is a technical specialist for the systems integration group at Infosys Technologies. He has around six years of experience in the information technology field as a developer and designer. His interests are in EAI and distributed computing.

abhinashak@infosys.com

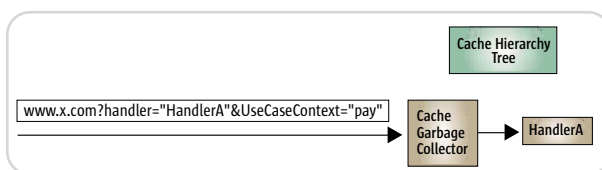


Figure 1 HTTP request processing flow

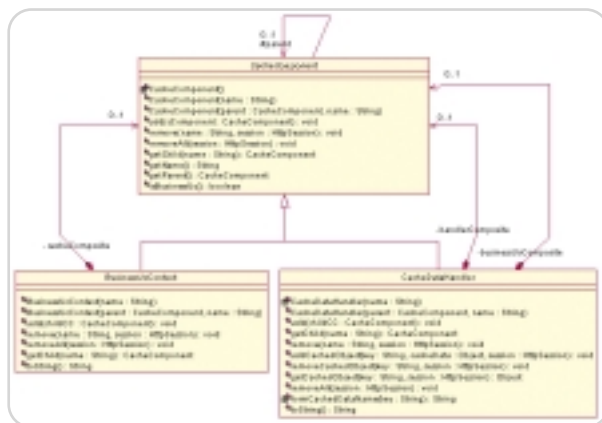


Figure 2 Cache hierarchy tree





# Monitored

The right approach to application life cycle management can transform your business.

## AllFusion™ Life Cycle Management Software

The key to great development isn't just great developers, it's great management. That's why we created AllFusion, a comprehensive application life cycle management solution. AllFusion has unprecedented flexibility that allows your projects to change along with the market. And that helps you do something a lot more important than just minimize aggravation. It lets you maximize productivity. To learn how to improve your development process, or to get a white paper, go to [ca.com/lifecycle](http://ca.com/lifecycle).



© 2003 Computer Associates International, Inc. (CAI). All rights reserved.



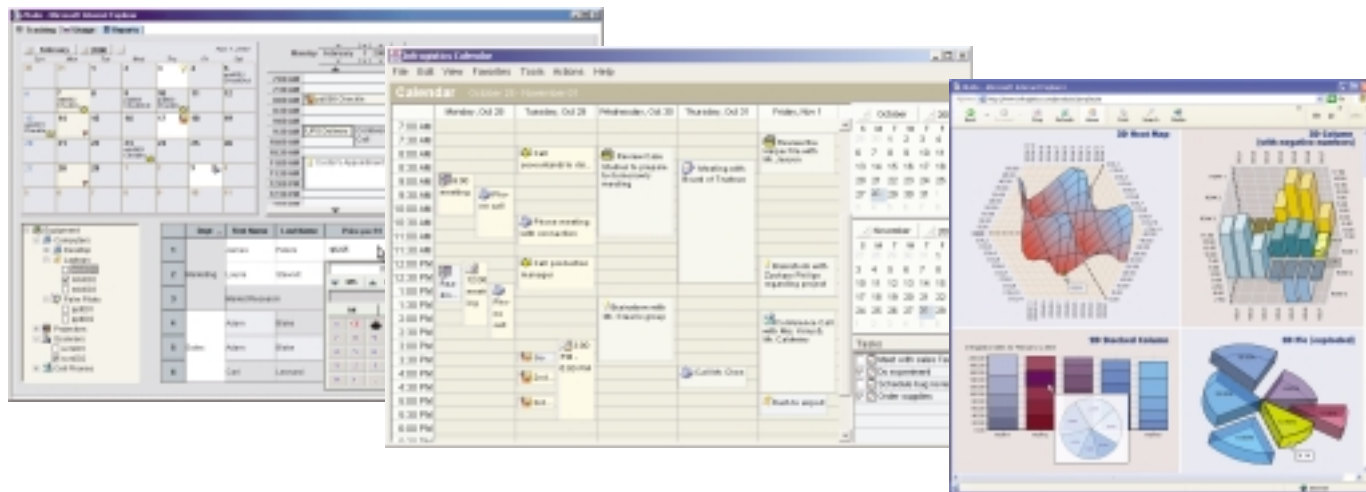
# Maximized





# Infragistics JSuite 7.0

## Tune up Your Presentation Layer!



### The Only Integrated Toolset for Presentation Layer Design

The Infragistics JSuite 7.0 is the only comprehensive framework of presentation layer components you will ever need for building visually superior front-end interfaces for any Java application. Create robust, eye-pleasing client- or server-side applications with the sophistication and usability of today's most recognized commercial applications.

**Data Models**  
**Tables/Grids**  
**Charting**  
**Trees**  
**Schedule**  
**Explorer UI**

**Gantt**  
**Editors**  
**Other GUI Controls**  
**Applet Wrappers**  
**Server Classes**

### AWT JFC JavaBeans™ Server

#### JSuite 7.0

##### Enterprise Edition

includes AWT, JFC, JavaBeans & Server product, subscription, as well as guaranteed priority support

è **includes ALL Java source code** **\$1495**

#### JSuite 7.0

##### with annual subscription

includes AWT, JFC, JavaBeans & Server a year's worth of updates, upgrades and new products

è **includes ALL Java source code** **\$995**

#### JSuite 7.0

##### product only

includes AWT, JFC, JavaBeans & Server **\$795**

Java Source Code is included in the JSuite 7.0 with Subscription or Enterprise Edition!

### Version 7.0 includes ALL NEW client- & server-side charting featuring ONE core charting engine:

- Both client- & server-side charts feature the same rich look-and-feel and object model.
- 2D/3D chart types - All major chart types: bar, stacked bar, column, stacked column, line, area, pie, scatter, bubble, heatmap, candle, hi/low, and open/close financial.
- All charts are fully customizable and extensible.
- Unique session-based security for protecting rendered chart data, making any and all sensitive charts secure.
- Bound and unbound modes, robust aesthetic enhancements, advanced legends, labels and tooltips, client- & server-side events, and much, much more!
- All source code for Java components (Subscription and Enterprise Editions)
- Licensed per developer (multiple installs for non-concurrent use). No runtime or server deployment charges
- RAD customizers maximize productivity, minimize code
- Create robust server-based thin client applications for the web with virtually the same rich user experience found in thick client implementation
- Create user interfaces connected to Web Services using XML and SOAP for seamless application integration

**Download a free, full-featured trial version!**  
**Order online, or contact us!**

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

**800-231-8588**







“Middleware is history”



EXCLUSIVE...

# The End of Middleware

by Jonathan Schwartz

Executive Vice President, Software Group, Sun Microsystems

**T**he marketplace tells you that “middleware is everywhere” when all along it should wise up and recognize that “middleware is dead.” Because that’s the new reality of enterprise computing today, according to Sun’s software czar Jonathan Schwartz.

What’s more important: Running your business or integrating middleware?

Should be an obvious answer, right?

Then why is the marketplace spending so much energy wallowing in the history of “middleware is everywhere”? Habit. A habit to which thousands of IT professionals devote their lives. But integrating middleware to build one-off business systems is about to perish with the rise of shared services – the services you’d like to build once, then execute on behalf of all your business systems.

As an example, when’s the last time you hired someone? Remember what it was like getting them “badged” and into the company? You had to grant them access to payroll, benefits, a desktop login, e-

mail, the CRM, and forecasting systems, then assign them an office and a phone.

Most likely, your company built a unique provisioning mechanism for each of the systems I just cited. One for HR, one for the sales force, one for information security, and yet another for physical security. And then you likely created even more redundancy by building one set for your intranet employees and another for your Internet customer or supplier systems. That’s inefficient, and in the world of Sarbanes-Oxley, a real problem – who has access to what? And why did you build 17 different systems?

Because it looked like a good idea at the time.

The same is true for most services that now make far more sense in a shared environment, from portals (how many does your company have?), to e-mail and application services, down to clustering and Web servers. There’s no real utility in having multiples of these services, as Nicholas Carr would point out, where your implementation doesn’t generate a competitive advantage. How you authenticate users and provision them with access to your systems is an unlikely competitive advantage. So why build a one-off solution instead of relying on an integrated system?

Good question.

Our belief is that the vast majority of Web services are better run as shared services. What’s the holdup, then? When we looked into this a year ago, we found three challenges:

### 1. Roadmap sprawl

There's no coordinating force causing all the required elements of shared services (from authentication to portals, Web services to clustering) to coalesce around a common release, interoperability, or support matrix. So you have no choice but to build your own.

### 2. Pricing

Middleware pricing is anything but shared – per CPU, per identity, per mailbox, per portal, per cluster node – pricing opacity obscures the real savings in running shared services. And if you can figure it out, you probably can't afford them.

### 3. Licensing

The industry relies on “common access licenses,” often tripling prices for services that touch the Internet – that's clearly an obstacle to shared services.

So here's how we solved the problems:

#### 1. The rise of Sun's Java Enterprise System

Sun's Java Enterprise System offers all the basic components, from directory and identity management to Web services, even e-mail and clustering. All in a single deliverable, prequalified, tested, and supported on multiple platforms.

#### 2. Pricing goes to a \$100/employee subscription

Why buy software differently than how you buy office furniture – by the employee? If your workforce decreases, you pay

less, and vice versa. The ultimate in predictable, transparent pricing.

#### 3. Licensing – infinite RTU

If the distinction between the intranet and extranet is disappearing, so should the distinction in our licensing. So \$100/employee buys you the right to use (RTU) all these services on all systems. At infinite scale – once you've paid for your employees, your customers are free. Free. It's your software.

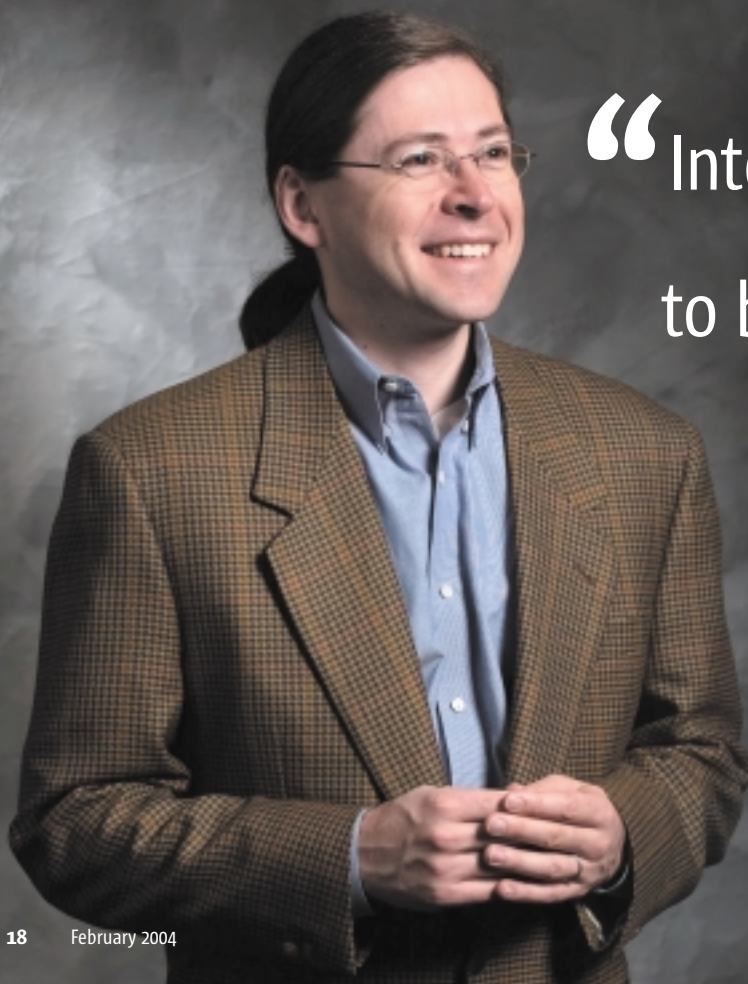
The vendors who believe hardware is commoditizing suggest the same forces won't affect software. We believe it affects both.

And as the world moves to recognize the value of a shared services infrastructure, it's our belief that middleware is history. Long live the system. The Java Enterprise System. ☺

---

#### Author Bio

As executive vice president of Sun's Software Group, Jonathan Schwartz spearheads the company's unified software business and focus and leads one of the largest software organizations in the industry. His market-leading group is responsible for the Solaris Operating System, defined by Sun as “the most secure, most scalable, most reliable operating environment on the planet”; the Java platform, which Schwartz and his team call “the gold standard for compatibility and security from the cellphone to the desktop to the datacenter”; a complete portfolio of highly integrated, highly affordable, interoperable, end-to-end software for every environment; the N1 operator platform (“for dynamic and utility computing”); developer tools; and Sun's entire family of middleware and software solutions. This essay was written exclusively for JDJ.



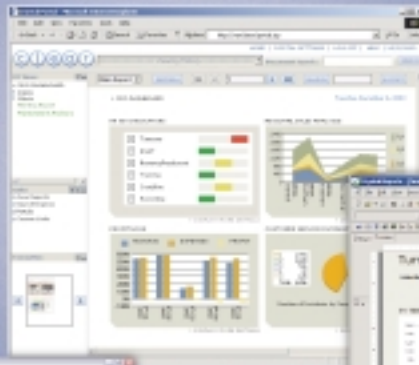
“Integrating middleware to build one-off business systems is about to perish with the rise of shared services”

99.9% of the world won't find these  
screen shots terribly exciting.  
But if you're in the other 0.1%,  
yeehaw.

Use Crystal Reports 10 with  
your J2EE applications }

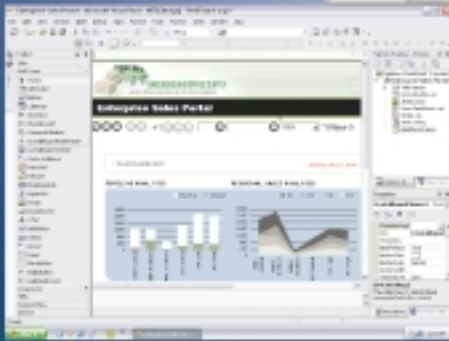
New 100% Java reporting component. Deploy  
reports across Unix, Linux and Windows

Extend Crystal Reports with Crystal Enterprise.  
Get world-class web report publishing,  
management, and scalability



Visual Designer simplifies  
data connectivity  
Deliver diverse data formatting  
options within your presentation layer

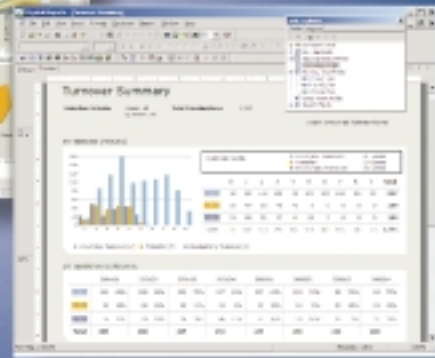
Access data natively, or via  
ODBC, JDBC and OLE DB



{ Expedite .NET and  
J2EE report integration

Design and integrate reports  
from within popular IDEs

Flexible Java, .NET and COM SDKs support the  
tight integration of report interactivity including:  
group tree navigation, exporting, printing, and  
drill down



## New Crystal Reports 10.

The best in business intelligence now offers the best in business reporting.  
New Crystal Reports® 10 is a faster and simpler way for developers to integrate dynamic  
data into applications and implement high-quality viewing, printing, and exporting. Learn  
more about Crystal Reports 10 and Crystal Enterprise™ 10, and access technical and evaluation  
resources at [www.businessobjects.com/v10/047](http://www.businessobjects.com/v10/047) or contact us directly at 1-800-877-2340.





# Go Wild Wirelessly with Bluetooth and Java

by Ben Hui

## Developing portable Bluetooth apps

**A**s a frequent visitor to J2ME and Bluetooth developer forums, I've noticed that one of the most frequently asked questions is "How do I get started with Bluetooth?" Despite its introduction in 1998 and a highly hyped year in 2001, Bluetooth application development remains hideous and challenging for lots of developers. Many experienced developers are looking for material that goes beyond a basic introduction and covers various aspects of real-life Bluetooth application development.

This article will take you to the next step of Bluetooth development and look at a newly developed specification that can assist Bluetooth developers in building applications rapidly. I'll discuss some important concepts of this specification and provide a walkthrough of a real Bluetooth application, BlueChat, that will cover some important design and implementation elements of a typical Bluetooth application. For those readers who have little or no experience with Bluetooth, the resources listed at the end of this article provide additional information.

To compile and execute the BlueChat example application, you need to obtain the J2ME Wireless Toolkit 1.0.4 and the Rococo Impronto Simulator. Refer to the resources section for the download location.

### Bluetooth Concept

Bluetooth is one of the wireless connectivity options available for mobile application development. It is characterized by a short range, low-power consumption; ad hoc networking; and usage-oriented design. Because of these characteristics, Bluetooth is often employed in consumer devices such as mobile phones and personal digital assistants (PDA). Typical usages of Bluetooth include answering voice calls using a wireless headset, synchronizing data between a PDA and a PC, and sending images from a camera phone to a PDA.

The functionality of Bluetooth is governed by a public and royalty-free specification and defined by the Bluetooth Special Interest Group. The Bluetooth specification defines both hardware and software layers. Traditionally, Bluetooth module manufacturers provide the necessary SDK to interface with their modules. One of the complexities of Bluetooth development is the fragmentation of these SDKs and APIs.

Vendor-specific SDKs force developers to adopt proprietary APIs for their respective Bluetooth chip sets. Bluetooth applications built on top of these APIs are not portable across devices and platforms. But isn't Bluetooth a standardized specification? The Bluetooth specification as defined by Bluetooth SIG is a functional specification of the technology. It describes how proper Bluetooth devices behave, and how they interoperate with each other (via Profiles). It does not describe how Bluetooth devices can be programmed nor how applications exercise a Bluetooth device's functions and utilize communication channels.

The need for a standardized API arises in order to develop Bluetooth applications in a platform-independent manner.

### Java Bluetooth API (JSR-82)

The Java Community Process introduced the first standardized API specification for Bluetooth back in 2000. This specification (JSR-82), Java API for Bluetooth Wireless Technology (aka JABWT), establishes a common ground for rapid Bluetooth application development. Developers are now able to write Bluetooth applications independent of hardware vendors. Most important, JABWT-compatible applications are portable across various JABWT-equipped devices.

The benefits of JABWT nevertheless come with a cost. The Bluetooth specification is designed to cover a diverse range of devices and usage scenarios. To complement this diversity the scope of JABWT took the lowest common denominator approach. Only the most commonly used profiles and functions are included in JABWT. In particular, Generic Access Profile, Service Discovery Profile, Serial Port Profile, Generic Object Exchange Profile, and their respective protocols are supported. These profiles allow JABWT applications to perform the following functionalities:

- **Generic Access Profile:** Provides the basic building blocks of a Bluetooth application, such as local device, remote device, Bluetooth address, and device discovery.
- **Service Discovery Profile:** Provides the ability to find available services to access remote functionalities.
- **Serial Port Profile:** Provides a stream-based connectivity between Bluetooth applications.
- **Generic Object Exchange Profile:** Provides support for OBEX protocol, which allows applications to exchange simple objects such as business card data.

To simplify the programming model, two entities, the Bluetooth Configuration Center (BCC) and the Service Discovery Database (SDDDB), are abstracted from the Java API. BCC includes the capabilities that globally configure the Bluetooth stack and prevent one application from adversely affecting another, typically a native application that exposes a user interface for users to parameterize the device. Although BCC is transparent to Java applications, it is important to realize that BCC has the ultimate authority over the Bluetooth host and may affect the behavior of your Bluetooth applications. SDDDB is an abstract database of service records, a collection of attributes that describe your Bluetooth services. Java applications interact with SDDDB indirectly via the update and retrieval of service records.

### Design of a Bluetooth Application – A Bluetooth Chat Room

To illustrate various aspect and design issues of a Bluetooth application, we'll develop a JABWT-based chat room application, called BlueChat, for mobile devices that must support the J2ME MIDP 1.0 profile. Users who have a



**Ben Hui** is a mobile technology specialist who develops software for PDAs and mobile devices and is keen on making J2ME technology work in harmony with daily life experiences. He has been programming with Java since its inception and has recently become addicted to consumer-device technologies.  
www.benhui.net  
contact@benhui.net

Compuware

# OptimalJ<sup>®</sup>



## THE POWER TO Develop, Transform, Reuse

Put your J2EE™ application development into overdrive using Compuware OptimalJ. This powerful model-driven, pattern-based enterprise development environment slashes the burden of repetitive coding, and helps you build high-quality applications that are truly reusable, time and time again.

Don't waste time writing boring infrastructure code. Start honing your skills now with this groundbreaking Java™ development tool.

**THE POWER IS RIGHT HERE.**

THE LEADER IN IT VALUE.

**COMPUWARE<sup>®</sup>**  
[www.compuware.com](http://www.compuware.com)



JABWT-capable device (such as the upcoming Nokia 6600 phone and Sony Ericsson P900) can use this application to chat with their nearby friends in an IRC fashion. When BlueChat launches, it searches and joins any existing chat room within the Bluetooth effective range, or creates a new chat room if it's the first active BlueChat in that range. We use the words chat room to represent a virtual chat room that's formed by a network of BlueChat applications. Users can start messaging with each other within the same virtual chat room when there's more than one party connected to each other. If one user sends a message over the air, all parties of the chat room will receive the message. Users can join and leave the chat room at anytime.

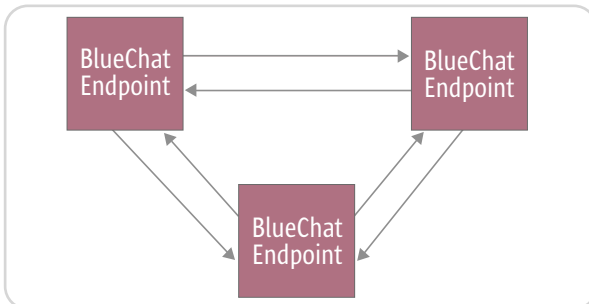


Figure 1 BlueChat topology

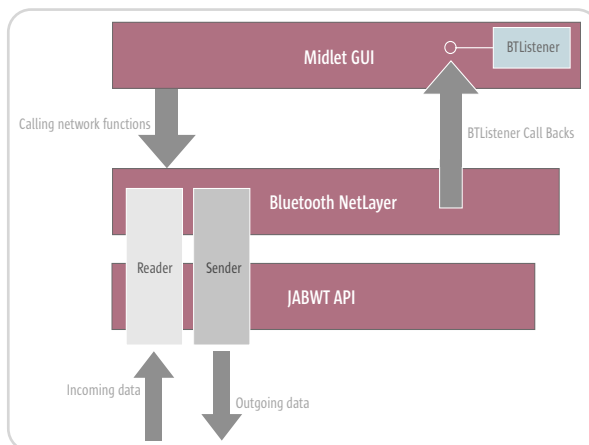


Figure 2 Network layer

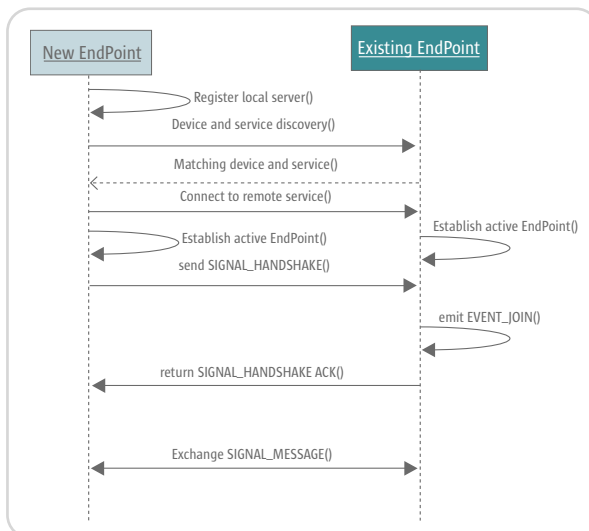


Figure 3 Establish two endpoints

For this article we make some assumptions to simplify the application in order to focus on Bluetooth implementation topics:

- There's only one chat room that exists within effective Bluetooth range.
- There is no security imposed when joining a chat room.
- Users run one instance of BlueChat on a device at any given time.

Before we dig into the source code, let's look at some of the Bluetooth application design issues. JABWT does a good job of providing a familiar API to J2ME developers for accessing Bluetooth facilities. JABWT is integrated with the J2ME Generic Connection Framework. As a result, Bluetooth network programming is very similar to a stream-based connection model.

Like many other network protocols, the Bluetooth connection model employs a client/server architecture. Our BlueChat application, on the other hand, operates in a peer-to-peer manner. Each running instance of BlueChat (or a node) can serve as a client and a server at the same time. It behaves as a client when BlueChat starts up; it searches and connects to existing running BlueChat devices. Once connected, it makes itself available for future clients to connect to. In such cases, it serves as a server for future client connections.

Figure 1 represents the network relationship between three BlueChat applications. To logically represent an active BlueChat node, we use the concept of endpoint to encapsulate all the connectivity attributes of a node. An endpoint represents a unique message delivery destination and source regardless of whether it is a server or a client.

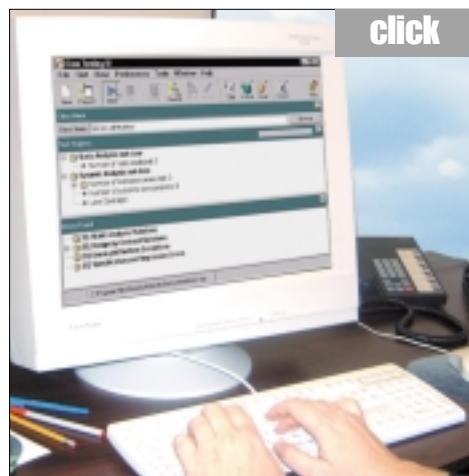
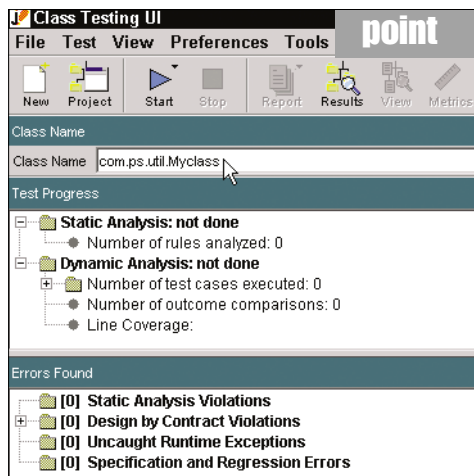
A Bluetooth connection differs from a regular socket connection by its unique device and service discovery processes. Bluetooth applications typically start the device discovery process to identify connectable devices, which is followed by a service discovery process to obtain a reference (URL) to suitable services. To hide these complexities from the Graphical User Interface (GUI) elements, a network layer is introduced to serve as a façade to the Bluetooth API. This design is comparable to the Model-Viewer-Controller model where the Viewer component is decoupled from the Model component. The GUI can access Bluetooth connectivity via a simplified interface, which does all the discovery and connection establishment behind the scenes. This network layer also provides the functionality to send messages to and receive messages from other endpoints. A call back interface is in place to report any network activity back to the GUI. Figure 2 illustrates the relationship between various layers and components in BlueChat.

The communication channel between each connected BlueChat endpoint is a structured data stream connection. We put together a simple protocol to coordinate the activity between each endpoint. This protocol includes the following features:

- **Initial handshake:** Each point must handshake with each other when the connection is first established. This ensures that the connecting device is a BlueChat node rather than a mistakenly connected application. During the handshake, we also exchange the screen names of the users (see Figure 3).
- **Delivery of text message:** Each sent text message is delivered to all endpoints connected to the BlueChat network.



# Automate unit test case generation for JUnit and Java™ with Parasoft Jtest.



**(It's as easy as 1-2-3.)**

**Parasoft Jtest is the first and only automated unit testing tool for Java™ development.**

With just a click, Jtest reads and analyzes code — quickly creating harnesses, stubs and test inputs — and tests without user intervention. Jtest also enables you to automate regression testing and static analysis. For loyal JUnit users, Jtest is designed to fully support existing test cases and automate the creation of new JUnit-compatible test cases.

**Learn how Jtest can enhance JUnit capabilities...**

Download a free eval copy of Jtest along with our informative new white paper entitled "Using Jtest With JUnit."

•----- **For Downloads go to [www.parasoft.com/jdj\\_02](http://www.parasoft.com/jdj_02). Or call 888-305-0041.**

Copyright ©2003 Parasoft Corporation. All rights reserved. All Parasoft product names are trademarks or registered trademarks of Parasoft Corporation in the United States and other countries. All other marks are the property of their respective owners.

**New for Java development...**

**Parasoft® Jtest® 5.0**

**Features:**

- Fully integrated into JUnit & Eclipse
- Quick Fix enabled finds & fixes errors fast
- Intuitively designed GUI

**Platforms:**

Linux  
Solaris  
Windows 2000/XP

**A part of Parasoft Automated Error Prevention (AEP) Solutions and Services**



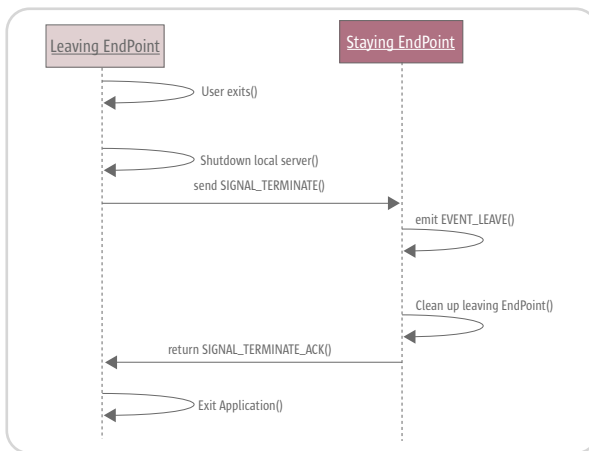


Figure 4 Terminate two endpoints

- **Termination handshake:** If the user quits the chat room gracefully, a termination token is sent to all the other endpoints to indicate its intention. We can clean up the necessary network and runtime resources associated with the leaving endpoint upon receiving this token. However, if the user walks away from effective range and becomes inaccessible, a termination token is not sent. Other active endpoints will discover the leaving party is inaccessible when the connections are lost, and they will clean up the resources (see Figure 4).

#### Implementation Consideration

The NetLayer class, which implements the BlueChat networking layer, does most of the Bluetooth-related work and provides the following functionality:

- Initializes the Bluetooth stack
- Registers BlueChat services to the Bluetooth device
- Searches for nearby devices
- Searches for BlueChat services on nearby devices
- Establishes endpoint connectivity for found BlueChat services
- Manages the life cycle of all endpoints

The Bluetooth stack can be initialized by calling `LocalDevice.getLocalDevice()`. `LocalDevice` is a singleton that uniquely represents the underlying Bluetooth device implementation. You can use the `LocalDevice` instance to gain access to other Bluetooth features including:

- Discovery agent (via `getDiscoveryAgent()`)
- Bluetooth physical network address (via `getBluetoothAddress()`)
- SDDB (via `getRecord()` and `updateRecord()`)

The BlueChat NetLayer's initial work is to create and register a BlueChat service to a local device. A Bluetooth service is an entry point for other Bluetooth clients to access available functionalities. Since each BlueChat endpoint can serve as a server, it must register its service in order to make this server available to other BlueChat clients. JABWT utilizes the MIDP Generic Connection Framework to instantiate a server connection. A BlueChat application needs to instantiate a Serial Port Profile connection, basically a stream-based connection that allows two BlueChat applications to exchange data using Java input and output streams. A BlueChat server connection is created using the code in Listing 1.

After a server connection is created, the service is not yet available to external clients (it is not discoverable). What has happened is that JABWT created a corresponding `ServiceRecord` for this service. A `ServiceRecord` is a collection of attributes that describes our service, and these attributes are searchable by clients. We can use `localDevice.getRecord( server )` to retrieve the newly created `ServiceRecord`. You may notice that the `ServiceRecord` is not empty at this point; it is already populated with some default values that are assigned by the JABWT implementation based on the connection string and the implementation configuration when we perform `Connector.open()`.

The `server.acceptAndOpen()` method notifies the Bluetooth implementation that the application is ready to accept incoming connections and make the service available. This also instructs the underlying implementation to store the `ServiceRecord` object in the SDDB, which occurs when `server.acceptAndOpen()` is first invoked. Notice that only the attributes stored in the SDDB can be seen and queried by other Bluetooth clients. Any subsequent change to the `ServiceRecord` must be reflected in the SDDB by using `localDevice.updateRecord()`.

#### Listing 1: Creating a server connection

```

// BlueChat specific service UUID
private final static UUID uuid = new UUID(0x6600BC);

...

StreamConnectionNotifier server = null;

// Create a server connection object, using a
// Serial Port Profile URL syntax and our specific
// UUID (0x6600BC)
// and set the service name to BlueChatApp
server = (StreamConnectionNotifier)Connector.open(
    "btspp://localhost:" + uuid.toString()
    +";name=BlueChatApp");

// Retrieve the service record template
ServiceRecord rec = localDevice.getRecord( server );

// set ServiceAvailability (0x0008) attribute to
// indicate our service is available
// 0xFF indicate fully available status
// This operation is optional
rec.setAttributeValue( 0x0008, new DataElement(
    DataElement.U_INT_1, 0xFF ) );
// Print the service record, which already contains
// some default values
Util.printServiceRecord( rec );
// Set the Major Service Classes flag in Bluetooth
// stack.
// We choose Object Transfer Service
rec.setDeviceServiceClasses(
    BluetoothConstants.SERVICE_OBJECT_TRANSFER );
  
```

#### Listing 2: Start discovering near-by devices

```


// initialize the JABWT stack
LocalDevice device = LocalDevice.getLocalDevice(); //
// obtain reference to singleton
device.setDiscoverable( DiscoveryAgent.LIAC ); // set
// Discover mode to LIAC
DiscoveryAgent agent = device.getDiscoveryAgent(); //
// obtain reference to singleton

// although JSR-82 provides the ability to lookup
// cached and preknown devices, we intentionally by-
// pass
// them and go to discovery mode directly.
// this allow us to retrieve the latest active
// BlueChat parties
//
// Listener object implements DiscoveryListener
agent.startInquiry(DiscoveryAgent.LIAC, new
Listener() );
  
```

#### Listing 3: Filtering relevant devices into pending EndPoint list

```

/**
 * A device is discovered.
 * Create a EndPoint for the device discovered and put
 * it on the pending list.
 * A service search will happen when all the qualifying
 * devices are discovered.
 */
public void deviceDiscovered(RemoteDevice remoteDevice,
    DeviceClass deviceClass)
  
```

A black and white photograph of an Apollo 17 scientist-astronaut on the moon. The astronaut is wearing a full space suit and is using a long-handled tool to retrieve a lunar sample from a rock. The moon's surface is rocky and covered in dust. The background shows the dark sky of space.

your mission depends  
on having the right tools.

Apollo 17 Scientist-astronaut retrieving lunar samples, December 1972

## ExtenXLS, the right tool for complex reporting™

### ExtenXLS 3, the 100% Java reporting toolkit

When HP, Ford, and Qwest Communications selected a solution for their mission critical reporting projects, they turned to ExtenXLS. With unparalleled support for complex Excel files, ExtenXLS enables creation of feature-rich Excel reports with charts, formatting, text, macros, formulas, and numerical data from any data source. Combining the flexibility of Java with the functionality and ubiquity of Excel, take a giant leap forward in reporting capabilities without abandoning existing business logic.

**Best-in-class Excel compatibility**, robust features, and a company dedicated to its products. ExtenXLS, the right tool for complex reporting.

Download a free evaluation copy today: [extentech.com/jdixls/](http://extentech.com/jdixls/)

**EXTENXLS<sup>3</sup>**  
JAVA | XLS REPORTING TOOLKIT™



- ✓ Fully supported
- ✓ Commercial-grade Excel compatibility
- ✓ Streamlined, easy to use API
- ✓ In-memory Formula Calculation
- ✓ 130+ Formula Functions Supported
- ✓ Best-in-class 3D Named Range support
- ✓ Add hyperlinks to Cells for drill-down reports
- ✓ Serialize & copy Sheets between WorkBooks
- ✓ Total control of fonts and cell formatting
- ✓ Comprehensive sample code and templates
- ✓ Used worldwide by global 1000 companies

 extentech™

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



Now our BlueChat application is ready to accept a connection. But what if your friends are already chatting prior to the start of your BlueChat? If there is an existing chat room available, BlueChat should join the existing network by searching for other BlueChat services on each individual device and connecting to their services. Three steps must be taken to perform this action.

1. Search for an available device.
2. For each available device, search for available and matching services.
3. For each available and matching service, connect to the service and perform the initial handshake.

DiscoveryAgent, another singleton in JABWT, can help us find other devices and services. Listing 2 shows the steps to search for devices.

There are two other options for retrieving connectable devices, a cached devices list and a preknown devices list. Cached devices are remote devices that have been discovered in a previous inquiry. Preknown are remote devices that are preconfigured in BCC. In our example, we choose to ignore both cached and preknown devices. We want to retrieve the most up-to-date list of active BlueChat devices at the moment BlueChat is launched. Therefore, our BlueChat application always initiates a new search for all surrounding devices.

Devices can be searchable in two modes, General Inquiry Access Code (GIAC) and Limited Inquiry Access Code (LIAC). When a device is set to GIAC, it basically means “I want to be discovered all the time.” Devices that provide public and per-

manent services fall into this category. Printers and fax machines are examples of GIAC devices. On the other hand, LIAC discovery mode means “I want to be discovered for a short period of time, as requested by my user.” Devices that provide on-demand connectivity will fall into this category. Examples are multiple player game consoles, mobile modems, and our BlueChat program.

The device discovery and service discovery processes are performed in an asynchronous manner. A Bluetooth application must provide a callback object for the JABWT implementation to notify when devices or services are found. This callback object (the NetLayer\$Listener inner class object in our case) implements the DiscoveryListener interface. When a device is found, the deviceDiscovered() method is invoked. We do some basic filtering (using the Major Service Class flag) to narrow down the candidate devices for our BlueChat application and ignore other unrelated devices (see Listing 3).

When all candidate devices are discovered, the device search is completed and the searchCompleted() method is invoked. We initiate the service discovery process using DiscoveryAgent .searchServices(). This is where the ServiceRecord attributes become useful. ServiceRecord is not only a description of the services, but also a query of constraints during service discovery. The second parameter of searchServices() allows us to specify which attributes and values the services must have in order for us to discover them. We can provide the UUID for the service that we registered earlier and it narrows down the exact

```

{
// only device of SERVICE_OBJECT_TRANSFER service
// will be considered as candidate device
// because in our BlueChat service, we explicitly set
the service class to
// SERVICE_OBJECT_TRANSFER. see the run() method
if ( (deviceClass.getServiceClasses() &
BluetoothConstants.SERVICE_OBJECT_TRANSFER) != 0 )
{
try
{
// create a inactive EndPoint and put it on the
pending list
EndPoint endpt = new EndPoint(NetLayer.this,
remoteDevice, null);

pendingEndpoints.addElement( endpt );

} catch (Exception e)
{
}
}
}
}

```

**Listing 4: Start discovering BlueChat services**

```

/**
 * device discovery completed.
 * After device inquiry completed, we start to search
for BlueChat services.
 * We loop through all the pending EndPoints and
request agent.searchServices
 * on each of the remote device.
 */
public void inquiryCompleted(int transId)
{
//
// for each EndPoint, we search for BlueChat
// services, i.e. ServiceClassIDList (0x0001) = uuid
(0x6600BC)
for (int i = 0; i < pendingEndpoints.size(); i++)
{
EndPoint endpt = (EndPoint) endPoints.elementAt(i);

//
// searchServices return a transaction id, which

```

```

we will used to
// identify which remote device the service is
found in our callback
// listener (class Listener)
endpt.transId = agent.searchServices(
new int[] {0x0001}, // attribute ID for ServiceClassIDList
new
UUID[] {uuid} // BlueChat service UUID

endpt.remoteDev,

new
Listener());
}
}

```

**Listing 5: Establishing connection to discovered BlueChat services**

```

/**
 * a service is discovered from a remote device.
 * when a BlueChat service is discovered, we establish
a connection to
 * this service. This signal joining the existing vir-
tual chat room.
 */
public void servicesDiscovered(int transId,
ServiceRecord[] svcRec)
{
if ( svcRec.length > 0 )
{
// We make an assumption that the first service
is BlueChat. In fact, only one
// service record will be found on each device.
// Note: we know the found service is BlueChat
service because we search on
// specific UUID, and this UUID is unique to us
String url = svcRec[0].getConnectionURL(
ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false );
StreamConnection con =
(StreamConnection)Connector.open( url );

// Establish active EndPoint
// and start sending SIGNAL_HANDSHAKE
...
}
}

```

matching candidate services on a remote device. This mechanism not only improves the performance of the discovery process, but also reduces the possibility of conflict. Once the desired service (BlueChat service) is found, we can retrieve the corresponding connection URL and establish the physical connection (see Listing 4).

To further validate that the connected service is indeed a BlueChat service, we immediately perform a handshake with the other party by sending a handshake signal (SIGNAL\_HANDSHAKE) and exchanging the user screen name. Receiving parties must respond with an acknowledgment (SIGNAL\_HANDSHAKE\_ACK) to confirm the request (see Listing 5).

To logically represent all the parties in the chat room, we introduce class EndPoint. From the application-level perspective, an endpoint encapsulates information for each actively connected BlueChat user and device. BlueChat uses EndPoint to identify which user to send a message to, and from which user a message is received. This abstraction allows us to hide the JABWT complexity from the GUI application. Endpoints are created when a connection is established between two BlueChat devices. Once created, we attach a reading thread and sending thread to the endpoint to manage the traffic between two endpoints. From this point on, two endpoints exchange user-entered messages (using SIGNAL\_MESSAGE) until a termination signal is received. Implementation of this protocol can be found in the Reader and Sender classes.

When a user exits BlueChat, the application sends the last message – a termination token (SIGNAL\_TERMINATE) – to all connected parties. This token signals that the endpoint is no longer active. All receiving parties must return an acknowledgment (SIGNAL\_TERMINATE\_ACK) and remove the leaving endpoint from the active endpoint list. An endpoint can also be removed when the connectivity is dropped, which suggests the user has left the chat room without an explicit exit command (possibly due to a user's walking away from the Bluetooth effective range).

Our GUI, based on the MIDP LCDUI API, provides a simple interface to send and receive messages. All received messages from all connected users are displayed sequentially on the screen, which creates a virtual chat room environment. When there are more messages to display than can fit onto one screen, older messages will roll off the upper edge. In this example application, users are not able to scroll back to see the past messages. Pressing the "Write" command takes users to a message-editing mode. Pressing the "Send" command sends the currently entered message to the chat room; all other connected users are able to see the message. To quit the chat room, pressing the "Exit" command sends a termination token to all other parties.

## Conclusion

Java/J2ME is the first platform to introduce a standard Bluetooth API, the Java API for Wireless Bluetooth (JAWBT). With the emergence of some JAWBT-enabled devices, developing portable Bluetooth applications becomes a reality for J2ME developers. In this article, we developed a Bluetooth application that allows us to understand the essence of Bluetooth development from design to implementation. Thanks to the J2ME-friendly API, experienced J2ME developers will find Bluetooth programming a familiar exercise. Device and service discoveries are important additions to the Bluetooth programming model, and should be fully understood by all Bluetooth developers.

One aspect of Bluetooth development not touched upon in this article is Bluetooth security. Bluetooth provides basic security features for authenticating and authorizing access to devices and services. I recommend reading Chapter 8 of the JABWT specification for details on how to build a secure Bluetooth application. ☺

## Resources

- *Java APIs for Bluetooth Wireless Technology (JSR-82) Specification:* <http://jcp.org/en/jsr/detail?id=82>
- *Benhui.net Bluetooth developer resources:* [www.benhui.net/bluetooth](http://www.benhui.net/bluetooth)
- *JABWT discussion group at Yahoo:* <http://groups.yahoo.com/group/JABWT>
- *The official Bluetooth member site:* [www.bluetooth.org/](http://www.bluetooth.org/)
- *Rococo Simulator Developers' Corner:* [www.rococosoft.com/devcorner/index.html](http://www.rococosoft.com/devcorner/index.html)
- *PaloWireless Bluetooth Resources Center:* [www.palowireless.com/info/tooth/knowledge-base.asp](http://www.palowireless.com/info/tooth/knowledge-base.asp)
- *Sun J2ME Wireless Toolkit:* <http://java.sun.com/products/j2mewtoolkit/>

# Advanced MVC Seminar

April 3rd, NYC

register @  
[basicportal.com/nyc](http://basicportal.com/nyc)



**Ted Husted**  
Struts fame  
[husted.com](http://husted.com)

**Rod Johnson**  
Author  
[springframework.org](http://springframework.org)

**Marc Canter**  
Founder of Macromedia  
[blogs.it/0100198](http://blogs.it/0100198)

**Howard Lewis Ship**  
Tapestry/HiveMind  
[javatapestry.blogspot.com](http://javatapestry.blogspot.com)

**Matt Raible**  
Displaytag/StrutsMenu  
[raibledesigns.com](http://raibledesigns.com)

**Clinton Begin**  
iBatis  
[ibatis.com](http://ibatis.com)

**Vic Cekvenich**  
basicPortal  
[basicportal.com](http://basicportal.com)

# The Delegation-Managed Persistence Entity Bean

*A composite entity bean for a new generation*

by Tal Cohen

**W**ith the introduction of the EJB 2.0 specification, the classic composite entity bean design pattern became outdated overnight. In this article, I present a new pattern that can serve as a proper replacement. This pattern, called Delegation-Managed Persistence bean (DMP bean), allows developers to represent objects that span multiple database tables. DMP beans provide a better solution than the original Composite EJB pattern without making you roll your own persistence mechanism.

Let's start with the basics. Why are composite entity beans required, anyway? The problem is that in many cases (depending on your application server and database), beans with container-managed persistence (CMP entity beans) can span only a single database table. However, in many enterprise databases, a single conceptual object is stored in numerous independent tables. The standard practice for solving this, in the days of EJB 1.x, was using bean-managed persistence (BMP entity beans). With BMP beans, the developer provides his or her own implementation for storing the bean to, and loading it from, the persistent storage (namely the database). This is a tiresome, repetitive, and error-prone job, and it often sacrifices portability for the sake of performance. Portability, for example, could be lost if nonstandard SQL statements (or even stored procedures) are used. The old composite entity bean pattern was basically just a bean that knew how to load itself from several tables using as many JDBC queries as needed, and likewise knew how to store itself to these tables, again using JDBC. This allowed developers to bypass the CMP limit of one table per bean, while providing a proper object-oriented representation of the notion represented by the bean.

A simple example would be the notion of client in an enterprise application, where information about each client is stored in several distinct tables – one table contains contact information, another contains the client's credit status, and so on. From an object-oriented point of view, we're interested in a single Client class that provides access to all the information stored in all the individual tables.

With the introduction of the EJB 2.0 standard, the common solution to such problems became using container-managed relationships (CMRs). A CMR allows a CMP entity bean to maintain a "relationship" to other CMP beans as long as these relationships are represented in the underlying database as foreign keys. This allows you to define fine-grained entity objects, rather than coarse-grained composite beans. The main problem with fine-grained objects in EJB 1.x was the price of remote method invocations. But now that entity beans are strongly encouraged to use the new local interface feature, this becomes a nonissue. Application clients access session beans, following the Session Façade design pattern, and these session beans access the entity beans using their local home and component interfaces.

This sounds like a good solution, but it suffers from two serious drawbacks. First, as noted earlier, the use of CMRs limits the usability of this solution to those composite objects that, in their database representation, use foreign keys. While this is indeed common, it is not always the case. The second problem is more bothersome: the fine-grained entities provide an accurate depiction of the database tables, rather than a high-level, object-oriented view of the concepts with which we deal.

True, the application clients do not deal with these low-level objects, but rather with high-level services offered by the session beans; but this is simply a shift of focus. The session beans now serve as clients to the entity beans. These session beans often contain key parts of the application logic – and it's expressed using a low-level view of the object model. This is unsatisfactory, and in fact contradictory to the original notion of entity EJBs as a high-level object model for the application data.

The original Composite Entity bean pattern solved this problem by providing a high-level view of the data, but at a great price, namely the tiresome and error-prone work required to create these beans. So allow me to present the new Delegation-Managed Persistence (DMP) bean pattern, which provides the same functionality and high-level view as composite entity beans do, while being easy to create and maintain, and taking full advantage of container-managed persistence.

The new pattern will be described here using the simple case of a composite bean Item, which is stored across two database tables: ITEM\_DATA\_1 and ITEM\_DATA\_2. We'll assume that each Item occupies one row in each of these two tables. Naturally, the pattern is applicable to a much wider range of cases.

Tal Cohen is a consultant specializing in J2EE and related technologies. Until recently, he worked as a researcher in IBM's Haifa Research Labs.  
tal@forum2.org



# Get the complete picture



## Features, Performance and Control

### Discover the ILOG JViews Graphics Components

You're developing a sophisticated user interface for a desktop, applet or servlet application – it needs to provide displays that go far beyond what Swing and HTML offer. How can you be sure it will have the features, performance, customization and scalability to enable your end-users to make better more informed decisions, faster?

With ILOG JViews, you get comprehensive graphical libraries & tools, resources, and maintenance services so you can focus on the implementation, confidently completing your application in less time and at less cost.

Quickly and easily build:

- Gantt and resource displays
- Graph layouts, diagrams, workflows
- Geographic map displays
- Realtime data charts
- Custom monitoring and control screens
- Network and equipment management screens

**Get a JViews Info Kit – Learn more, test drive an Eval.  
Go to: [jviews-info-kit.ilog.com](http://jviews-info-kit.ilog.com) or Call: 1-800-for-ILOG**



Changing the rules of business™

We begin by defining two CMP entity beans, `ItemData1` and `ItemData2`, over the two database tables. These are the DMP bean's underlying fine-grained entities, and we'll refer to them as Item's component beans.

Next we define the `Item` class as a BMP entity bean. Don't worry, while defined as a BMP bean, our DMP bean will not really have to manage its own persistence issues.

In the bean class, we define a field for each of the component beans: these fields are the component bean references and `Item` has two such fields. We also define the component key references as one additional field per component bean; the type of these fields is the type of each component bean's primary key class.

Again for the sake of simplicity, we will assume that both `ItemData1` and `ItemData2` use `java.lang.Integer` for their primary key classes. So `Item`'s bean class definition begins like this:

```
public class ItemBean implements EntityBean {
    // Component bean references:
    private ItemData1Local itemData1;
    private ItemData2Local itemData2;

    // Component key references:
    private Integer itemData1Key;
    private Integer itemData2Key;
```

Note that, true to the spirit of EJB 2.0, we access the underlying fine-grained entities via their local component interfaces. As you can probably guess, the component key references are maintained so we can lazily load the beans on a per-need basis. Two private methods, `getItemData1()` and `getItemData2()`, will be used internally to access the component bean references. The pseudo-code for the first of these methods would be:

```
private ItemData1Local getItemData1() {
    if (itemData1 == null) {
        // find local home, probably using a home factory
        ItemData1LocalHome home = ...;

        itemData1 = home.findByPrimaryKey(itemData1Key);
    }

    return itemData1;
}
```

In itself, `Item`'s bean class does not contain any fields for representing bean attributes. Any getter or setter method for loading or changing attribute values is delegated to the underlying component beans via the component references, like this:

```
public getSomeAttribute() {
    return getItemData1().getSomeAttribute();
}
```

Like attributes, any business operations defined in the component beans can also be made available in the higher-level DMP bean using delegation. Yet we can also provide new, more complex features that involve accessing several attributes or several business methods from one or more of the component beans. We are, in effect, providing a high-level view of the single business notion stored across multiple database tables.

### The Primary Key Class

The primary key for a DMP should be a composite key: a simple Java class that includes (as fields) the primary keys for every component bean class. In our example, the `ItemKey` class would have two fields of type `Integer`, one for `ItemData1`'s key and the other for `ItemData2`'s. All normal primary key rules apply here – make sure the class is serializable, has proper `equals()` and `hashCode()` methods, and so forth. Getters and setters for the internal keys are also in order.

### Loading and Storing DMPs

In most BMPs, the key life-cycle methods – `ejbLoad()` and `ejbStore()` – are complex beasts, accessing one or more tables in the database directly by means of JDBC. Surprisingly, in DMP beans, `ejbStore()` is completely empty, while `ejbLoad()` is extremely simple and involves no manual database access.

No implementation is required for `ejbStore()` since any update is delegated to the component CMPs, which by their very nature allow the container to manage their persistence needs.

As for `ejbLoad()`, in this method the bean obtains its composite primary key from its entity context object and checks if any of the internal keys is different from the privately maintained component key references. If a component's key was changed, we must store the new value, and we can no longer assume that the local reference to that object is valid. To invalidate the maintained reference, we simply nullify it, and it will be loaded again when needed due to the lazy evaluation mechanism detailed earlier. So in our example, `ejbLoad()` would look like Listing 1.

While technically a BMP, the composite bean does not manage its own persistence: it indirectly delegates it to the container. This is why it was named “Delegation-Managed Persistence bean” or “DMP bean” in the first place.

### Passivation and Activation of DMP Beans

No special actions are required when DMP beans are passivated or activated. Still, it could help the container better manage its resources if the component references and component key references are all nullified in `ejbPassivate()`.

### Creating, Finding, and Removing DMP Beans

Perhaps the most sensitive part of this pattern is the implementation of the `ejbCreate...()` and `ejbFind...()` life-cycle methods. The last remaining life-cycle method, `ejbRemove()`, is rather simple to implement: just remove each of the component CMPs in turn. Make sure `ejbRemove()` has the `REQUIRES` transaction attribute, so if the removal of any of the component beans fails, no removal will take place.

Each `ejbFind...()` method should return the composite primary key type. This is basically done by finding each of the relevant component CMPs (via whatever finder methods they provide in their own local home interfaces, and possibly using CMRs between these CMPs), and then composing the required primary key from its components (the primary keys of the component CMPs).

Slightly more complex is the case of finders that return collections. While it is possible to retrieve the relevant collections of composing keys, and then iterate over them in order to create a new collection of composite keys, this could be highly ineffective. One solution is to create a lazy collection mechanism that keeps the collections of composing keys, and delves into them only when an iterator is

used (a lazy collection). Bear in mind, however, that (depending on your application server software) collections returned by local home interfaces of CMP beans are often lazy collections themselves, and are invalidated as soon as the transaction that created them is over. In such cases, there is no option other than to create the whole collection immediately inside the DMP bean's finder method.

But DMP beans can do more than rely on the finder methods provided by their composing beans. The finder methods are one place where it does make sense to use JDBC directly in DMP beans. Using raw SQL, you can create finders that are too complex, ineffective, or downright impossible to implement using EJB QL. Thus, if your high-level business notion, which spans multiple database tables, suggests high-level search criteria that cannot be expressed by simple searches on individual tables, you can make these searches available without manually iterating over collections of fine-grained objects.

Finally, as for `ejbCreate...()` methods, these should create the relevant component CMPs (using their own `create...()` methods from the local home interface), and maintain the resulting local references in the DMP bean's fields. Again, as with `ejbRemove()`, the entire creation process should normally be enclosed within a single database transaction.

## Conclusion

The pattern presented here allows developers to easily create a high-level object-oriented view of complex business objects, which cannot be represented using CMP

entity beans due to mapping limitations. These high-level objects can then be used by their clients (normally session beans that would access them via a local interface), simplifying the client code since it no longer has to be aware of the internal structure of these potentially complex objects. While overcoming the limitations of CMP entities, Delegation-Managed Persistence Beans do not necessitate the creation of complex persistence code, since they take full (if indirect) advantage of the automated persistence services offered by the container. ☺

### Listing 1

```
public void ejbLoad() {
    ItemKey myKey =
        (ItemKey) getEntityContext().getPrimaryKey();
    Integer newKey1 = myKey.getData1Key();
    Integer newKey2 = myKey.getData2Key();

    if (!(itemData1Key.equals(newKey1))) {
        itemData1 = null;
        itemData1Key = newKey1;
    }

    if (!(itemData2Key.equals(newKey2))) {
        itemData2 = null;
        itemData2Key = newKey2;
    }
}
```

**True, good  
tech jobs are  
nearly impossible  
to find.**

Find them here.

**Dice™**  
**Tech Jobs. Tech Talent.**

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





Jason Bell

Core and Internals Editor

# Man with an Open Heart

I'm a firm believer in seasons of work for a specific job. The season of writing for me is coming to a nice close – this is my last editorial for *JDJ* (though I still have reviews that I have to get on with). It's been fun watching the Java world open up before me during the working day, blogging something, and then enjoying the feedback. I've enjoyed the feedback, the e-mails, and even the criticisms (thanks JP!). It's from these exchanges that I learn and learn some more. There's only one mistake and that's not learning from your mistakes.

There have been days when I look at all the work I've done and wonder if any of it was worth it. In the end, no work is wasted, just reused, relearned, and refactored into something better and more robust. This is the ever-continuing journey of the software developer: being a craftsman and getting a feel for crafting software. It's a discipline and it takes time to perfect. For me, personally, I don't want to be the Java celebrity; I want to be a Java craftsman. Even if it's just an interface file, I want it to be an interface of quality. Now the suffering for quality is usually not apparent to others; it's your journey so make of the situation what you will.

Sometimes people need a change in direction. Not so long ago I wrote about handing in my notice with no job to go to. I'm now about to start a new position that I'm getting really excited about. In fact, it feels like everything I have been learning will be poured out in this new position. If the trumpets sound, I'll let you know.

If what you are reading sounds like a load of waffle, you're entitled to your opinion. Once I was involved in blogging, both reading and writing. I've backed off a lot now. The main reason has to do with the personality of a blogger. Lots of them hide behind their blogs, using them as a smoke-screen. Those who have read Bileblog know to take it with a pinch of salt (though I respect what Hani says some of the time). Other blogs have just turned into a bitching session. It's one that I don't enter into. Over the past year I've become more guarded. I'm in a position where my words could be taken and misrepresented, and these positions cannot be abused. Just because you are writing a blog doesn't mean that your professionalism should be left at the door along with your shoes. Personality is a dangerous thing and

“I cannot stress the importance of your own personal development. Learning is a constant process. Whether you do J2EE or J2ME, there is still time to delve into the core API and cover areas that you may not normally cover”

For me this journey has been fun, painful, and humbling. The same goes for the craftsman in me. Don't be afraid to put your hand up and say you don't know something. I've done it many times. There are 23,000+ methods in the core Java API. Do you expect me to know them all? No, that's why the API documents exist. It's a journey that's supposed to be fun, as well as painful once in a while. The light in the distance is just waiting for you to run toward it and learn something new. This is what makes Java such an interesting journey for me.

As a musician it took a long time for me to accept that all those hours of practice, bleeding fingers, and painful cramps in my hands were for just one moment in April 2001. That night the heavens opened and the golden trumpets sounded, and it was like gold pouring in from above. All I was doing was playing guitar and singing, but it felt as if everything had led up to that moment, a moment I will never forget. I'm waiting for that Java moment. When the heavens will open again, all this work, learning and coding, will be used in something that will remain in my memory forever. No one can steal these moments away from you; hold onto them. It's fun. Nothing ever happens by accident. I often questioned Alan Williamson why he wanted me to be J2SE editor and he simply said, “There was just something.” I trust these moments of opportunity when crossroads appear. They have to be embraced and followed, as you don't know what is behind that door.

people's personalities can rub off on you very quickly indeed. I've become really picky about who I hang around with. I've given reading blogs a rest (with the exception of a few) and am concentrating on what I should have been concentrating on, coding in Java.

So, brothers and sisters who program Java, be true to yourselves. Code to a quality you'd expect to see from others. Be on hand to help. In *Software Craftsmanship* by Peter McBreen, he focuses heavily on the role of journeyman developers who train the apprentices and the masters who train the journeymen. Build your network of contacts. I use LinkedIn a lot and it's very helpful. At the end of the day you are only as good as the code you write.

Last, I cannot stress the importance of your own personal development. Learning is a constant process. Whether you do J2EE or J2ME, there is still time to delve into the core API and cover areas that you may not normally cover. For me it's been Swing and AWT. I'm a server-side developer so I've had no real use for GUI applications. I've been forcing myself to learn this stuff as I can just see a time when I'll need it. I've been playing with Bluetooth/J2ME development as well. I have a review coming up for the Sony Ericsson P900 smart phone, so this has been a good chance to learn something new. If you don't enjoy getting up in the morning, perhaps it's time to rethink where you're at. ☺

Jason Bell is a technical architect for a business intelligence company in England. He is also involved in a number of open source projects and reads the API docs.

jasonbell@sys-con.com

Download your free  
evaluation copy today at  
[www.oakgrovesystems.com/jdj](http://www.oakgrovesystems.com/jdj)

# Announcing Reactor 5.5!

# Welcome To Workflow Nirvana

*The Functionality You Want  
The Ease of Use Your Business Users Demand*

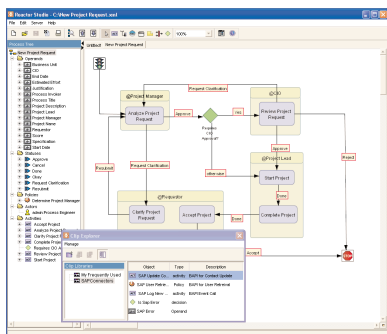


Life is in balance. All is in harmony. Developers are fulfilled. Business users are satisfied. And even cats and dogs have agreed to live in peace worldwide. Because at long last, Reactor 5.5 is here.

Already the ideal workflow engine for developers seeking control, extensibility, and platform neutrality, Reactor 5.5 is now the perfect tool for business users, giving them the ease of use they demand.

J2EE-based, XML-driven and available as royalty-free source code, Reactor 5.5 frees you to follow the path to true enlightenment (and killer gains in productivity) by providing:

- A single, scalable process platform for defining and linking enterprise-wide workflows
- Extensible libraries of re-useable process components
- A user-friendly tool for point-and-click development of custom workflow solutions.



Process design tool, Reactor Studio, enables quick development and deployment of workflows.

So light some incense, find your mantra and focus your chi. Because Reactor 5.5 is here to take you to Workflow Nirvana. Download a free evaluation copy at [www.oakgrovesystems.com/jdj](http://www.oakgrovesystems.com/jdj) or call us at (818) 440-1234.

We'd love to help you...

**Declare Your Workflow Independence!™**

**oakgrove**  
systems

Workflow Automation – Business Process Integration – Web Services Orchestration

# Java Collections

by David McReynolds

## Managing collections

As Jason Bell pointed out in his editorial “A Modern Day Cinderella” (*JDJ*, Vol. 8, issue 9), the spotlight is on J2EE and as a result many programmers are ignoring the foundation of the JDK. J2SE is the Java equivalent of C/C++ standard libraries. Here we deal with the lower-level entities, like the Number types, Integer, Long, Float, and Double.

The Java Collections Framework (JCF) should be your first choice when faced with the task of managing any type of collection. The Collections API is one of the most useful parts of the JDK. Looking back at the projects I’ve worked on over the past 13 years, to some degree all of them involved managing collections of data structures.

In this article I’ll review the collections architecture. I’ll also point out some of the useful features of the collections API (sorting and searching). To begin I’ll go over the class categories, followed by a more detailed explanation of each. I encourage you to review Sun’s documentation at <http://java.sun.com/products/jdk/1.2/docs/guide/collections/reference.html>.

There are explicit class categories in the Collections Framework. The J2SE Collections Framework consists of interfaces, abstract base classes, and concrete implementations that provide a rich set of functionality for us. The implementations are the classes your application should be utilizing behind the scenes. There are implementations based on maps and others that are backed by arrays. You can make your collection read-only or you can add support for multi-threaded access. How is a programmer to decide which entity to use? There are two main criteria: thread safety and usage semantics.

Usage semantics can be further broken down into collection- or map-based access. The library makes a distinction. The Map interface is not related to any of the Collection interfaces, because its main purpose in life is to map a key to an object, while the collection is just a loosely associated group of objects.

### Interfaces

Figure 1 provides a class diagram showing the interfaces that make up the Collections API. The interfaces represent the ideal types you should be passing around in your application.

I strongly urge you to expose only the interfaces to clients of your classes. If you don’t do this and instead pass around references to the concrete implementations, your code will become brittle due to the number of changes required to swap out one interface implementation for another. You should strive to expose the most general interface. For example, if a method is to return an ArrayList, first look and see if the methods exposed by the Collection interface will meet the needs of the intended usage (see Table 1). By doing this, you give yourself the opportunity to modify your method to return a LinkedList or any other type supporting the

Collection interface. Who knows? You may even want to provide your own implementation of a Balanced Tree, and if you are instantiating and passing around references to a TreeMap, you’ll have to alter the code at each reference.

### Sets

The semantics of Sets are close to those of Lists. However, Sets lack the notion of direct random access. A Set is just a collection of objects that you may iterate over. A useful feature of Sets is that they do not allow duplicates as long as you override hashCode() and equals() from Object. Listings 1–3 provide a short program that will illustrate this effect. There is the main HashSetExample and two Person classes: one that does not override Object.equals()/hashCode() and one that does.

Running this program produces the follow output.

```
[000-11-1111, 222-23-1234, 000-11-1111]
[000-11-1111, 222-23-1234]
```

To remove duplicates your classes must override equals() from java.lang.Object. According to the Javadoc, overriding Object.hashCode() has more to do with performance. Interestingly, Sun’s “Introduction to the Collections Framework Short Course” mentions overriding only the

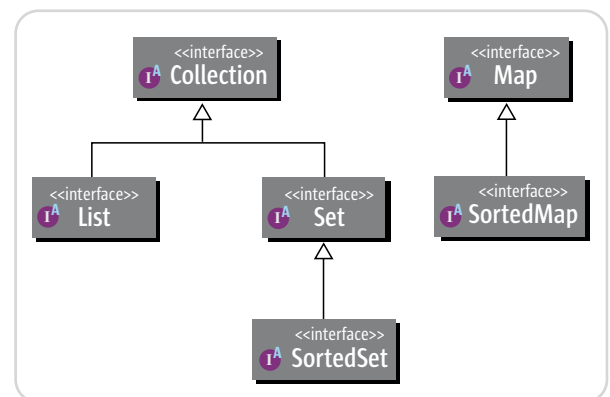


Figure 1 Collections Framework major interfaces

|            |            | Implementations |                 |               |             |
|------------|------------|-----------------|-----------------|---------------|-------------|
|            |            | Hash Table      | Resizable Array | Balanced Tree | Linked List |
| Interfaces | Collection | Set             | HashSet         | TreeSet       |             |
|            | List       |                 | ArrayList       |               | LinkedList  |
|            | Map        |                 | HashMap         | TreeMap       |             |

Table 1



David McReynolds

has been programming for over 12 years and is currently employed by Daugherty Business Solutions as a consultant. He has an MS in computer science from Southern Polytechnic State University.

david.mcreeynolds@daugherty.com



# MapObjects—Java Edition

## *Introducing a Pure Java Solution for Adding Dynamic Maps to Your Applications*

Imagine the possibilities of building custom applications that allow users to see information visually represented on a map. With such a tool, your users could spot patterns and trends in data that might otherwise go unnoticed, resulting in better, more informed decision making.

The ESRI® MapObjects®—Java Edition software is a suite of more than 900 Java developer components that you can use to build custom, high-powered geographic information system (GIS) applications or applets at the client, Web, and server tiers.

With MapObjects—Java Edition, you can

- ▶ Build applications with a wide range of mapping and GIS capabilities (labeling, panning, zooming, measuring distances, querying data, and much more).
- ▶ Easily drag and drop visual JavaBeans into an integrated development environment at design time.
- ▶ Use Swing Media components to quickly create a user-friendly interface.
- ▶ Combine multiple data sources.
- ▶ Display a variety of data types.
- ▶ Deploy J2EE server-based applications.

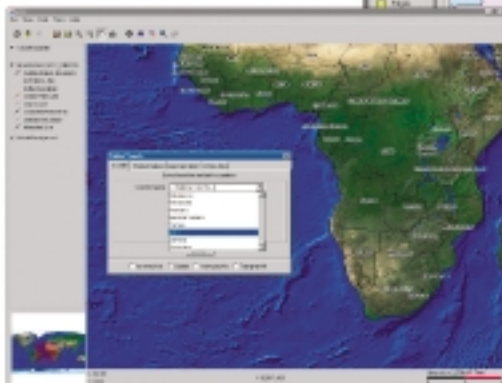
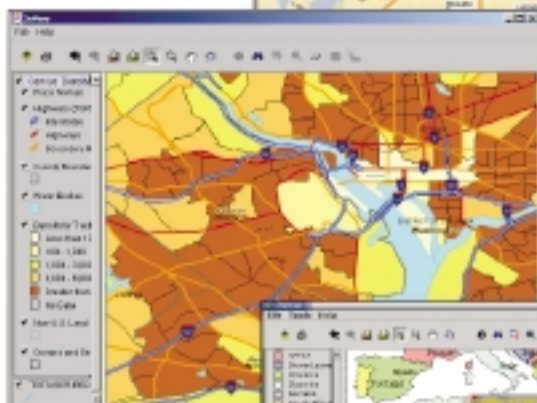
### **MapObjects—Java Edition**

*Mapping and GIS Tools for the Java Development Environment*



**1-888-332-2320**

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



Object.hashCode(). Beware that if you follow the tutorial to the letter, you'll still have duplicate entries. You must override Object.equals(), as I've done in Listing 2, to prevent duplicates in your Sets.

How about sorting this list? TreeSet can do that for us, but we still have a choice to make. Will we be sorting by the natural order or do we want an ad hoc sort? For this article we'll examine the ad hoc sort (to implement your own natural order, your class should implement the Comparable interface). We can achieve an arbitrary sort order by utilizing the Comparator interface. When we employ a comparator, it's passed to the sorting object. First, we need to create our sorting algorithm (see Listing 4).

Now we can give this algorithm to the other implementation of Set; TreeSet. We add the following code to our main method at line 29 in Listing 4.

```

29
30 Set sortedSet =
31     new TreeSet
32         (new PersonComparator());
33
34 sortedSet.addAll(set);
35
36 // sorted
37 System.out.println(sortedSet);
    
```

Now the output becomes:

```

[000-11-1111, 222-23-1234, 000-11-1111]
[000-11-1111, 222-23-1234]
[222-23-1234, 000-11-1111]
    
```

Cool, eh? I won't go over this for each implementation. You should be able to apply this concept to any of the other sorting containers or utility methods (from Arrays or Collections). It's worth pointing out that even if you don't override either method, the TreeSet will use the Comparator and eliminate duplicates in the sorted set. Figure 2 provides a class diagram for the Set category.

The LinkedHashMap is a special implementation of HashSet that supports list operations without directly implementing the List interface. LinkedHashMap will maintain the insertion order of the list elements, yet still allow you to access elements via a key, such as a traditional Map. And, as the name implies, it is a Set that supports all of Set's operations.

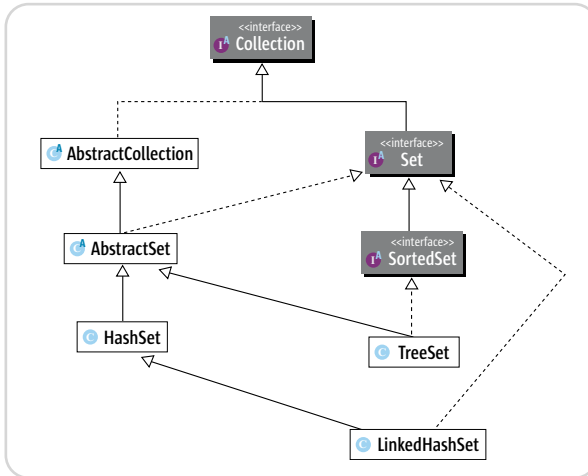


Figure 2 Set category

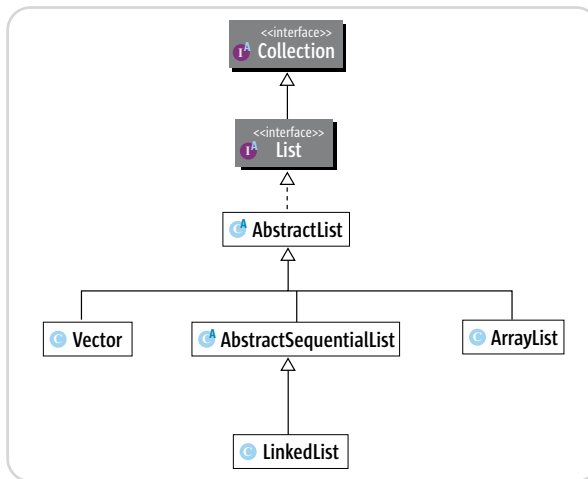


Figure 3 List category

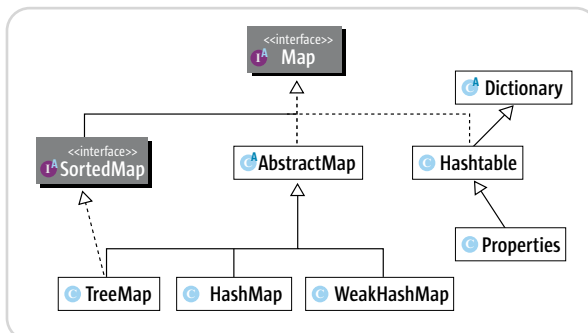


Figure 4 Map category

### Lists

Collection's other category is List; the implementations are ArrayList and LinkedList (see Figure 3).

The List interface supports the notion of direct index-based access to the entries, allows duplicates, and defines an order. Direct index-based access is realized via the get(int) method, which accepts an index as the only argument. You may even acquire a subset of the List by specifying a "from index" and a "to index", the semantics of which closely follow that of String. The element at "from index" will be included in the sublist while the element at "to index" will not.

ArrayList should be preferred if you don't require the ability to insert elements into the middle of the List (you're always adding to the end of the List) and you require random access to the elements. However, if you need to insert elements into the List and sequential access is your main concern, then LinkedList will be better.

### Maps

Finally, we get to the Map category (see Figure 4). As mentioned earlier, Map is not related to any of the Collection classes. This is because the JCF authors wanted to make a clear distinction between Collections and Maps. The most notable difference is that Maps do not support index-based access semantics. What is the *n*th element of a Map?

If a Map is a Collection, what are the elements? The only reasonable answer is "Key-value pairs," but this provides a very limited (and not particularly useful) Map abstraction. You can't ask what value a given key maps to, nor can you delete the entry for a given key without knowing what value it maps to.





<http://www.webappcabaret.com/jdj.jsp>

## J2EE Web Hosting

Quality Web Hosting at a reasonable price...

# <JAVA WEB HOSTING AND OUTSOURCING>

You have developed the coolest mission-critical application. Now you need to deploy it. Outsource your hosting and infrastructure requirements with WebAppCabaret so you can save time and money and concentrate on other important things. WebAppCabaret is the leading JAVA J2EE Web Hosting Service Provider. From shared hosting to complex multi-dedicated server hosting, WebAppCabaret has the right solution for you. 30 Day Money Back Guarantee and SLA. WebAppCabaret offers the latest Standards based Servlet containers, EJB servers, and JVMs. We provide options such as e-Commerce, EJB 2.x, Failover, and Clustering. Our Tier 1 Data Center ensures the best in availability and performance.

At WebAppCabaret you have the flexibility to choose the LATEST hosting technology best suited for your WEB application or your programming skill. If you are a programmer or consultant, WebAppCabaret has the right hosting solution for your project or client's dynamic web application/services requirements.

### OUTSOURCING:

Do you really need an IT department for your web applications, mail systems, and data backups when we can perform the same functions more efficiently at a fraction of the cost - with competent technical expertise and redundant hosting facilities.

### J2EE HOSTING:

Below is a partial price list of our standard hosting plans. (Reseller accounts also available). For more details please log on to <http://www.webappcabaret.com/jdj.jsp>

### \$39/mo Enterprise

Latest JBoss/Tomcat/Jetty  
Latest JSP/Servlets/EJBs  
Private JVM  
Choice of latest JDKs  
Dedicated IP Address  
NGASI Control Panel  
PHP and Perl  
Web Stats  
1GB Disk  
200MB DB  
MySql  
PostgreSQL  
Dedicated Apache  
Telnet . SSH . FTP  
5 Domains  
100 Emails  
Web Mail . POP . IMAP  
*more...*

### \$191/mo Dedicated

Managed Dedicated Server starts at \$191 per month for:  
256MB RAM  
Pentium 4  
40GB RAID  
Firewall  
Unlimited Domains  
Unlimited Web Site Hosting  
NGASI Control Panel with your own Logo  
Each dedicated server configured for standalone web application and web hosting (resellers) at no extra charge.  
*more...*

### \$17/mo Professional

Latest Tomcat/Jetty  
Latest JSP/Servlets  
Private JVM  
Choice of latest JDKs  
NGASI Control Panel  
PHP and Perl  
Web Stats  
200MB Disk  
30MB DB  
MySql  
Telnet . SSH . FTP  
2 Domains  
20 Emails  
Web Mail . POP . IMAP  
*more...*

### \$99/mo Reseller

If you cannot afford a Dedicated server for reselling web hosting, for \$99/mo the Shared Reseller plan gives you:

5 Professional Plans  
10 Basic Plans  
NGASI Control Panel with your own Logo  
50% Discount  
No System Admin  
Easy Account Mgt

*more...*



The workhorse of this category is HashMap. For inserting, deleting, and accessing elements, HashMap offers the best implementation. TreeMap is the sorted version and offers the ability to traverse the contents of the Map in a determined order.

As with the HashSet earlier, HashMap will require you to override Object.equals() and have a defined Object.hashCode() implementation on your own classes. And, of course, the objects you place in TreeMap should be comparable (or you must use the TreeMap(Comparator) constructor).

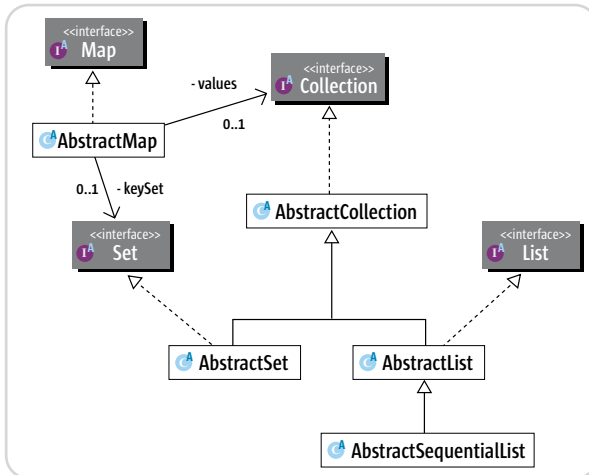


Figure 5 Collection abstractions

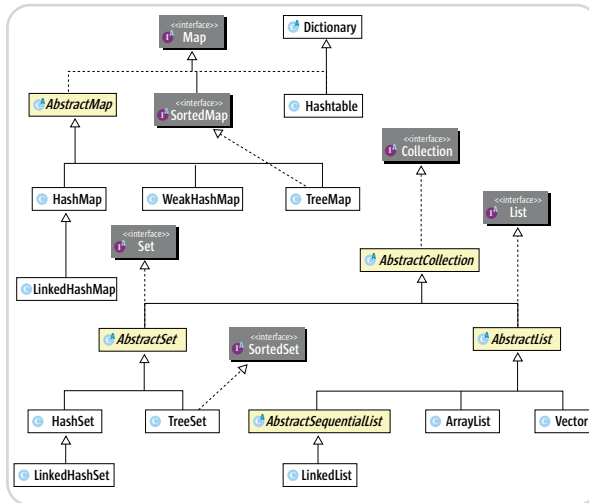


Figure 6 Implementation classes

| Compare Synch to Non-Synch with 100,000 Accesses |           |       |        |       |                   |       |     |
|--|-----------|-------|--------|-------|-------------------|-------|-----|
| Time(ms)   |           |       |        |       |                   |       |     |
| Type   | ArrayList |       | Vector |       | Synchronized List |       |     |
| Action   | Read      | Write | Read   | Write | Read              | Write |     |
| Test Run   | 1         | 20    | 80     | 70    | 351               | 30    | 250 |
|  | 2         | 20    | 60     | 70    | 381               | 20    | 250 |
|  | 3         | 40    | 60     | 71    | 350               | 30    | 240 |
|  | 4         | 30    | 51     | 70    | 380               | 50    | 260 |
|  | 5         | 40    | 60     | 70    | 371               | 20    | 250 |
|  | 6         | 20    | 80     | 71    | 340               | 20    | 291 |
|  | 7         | 20    | 60     | 71    | 350               | 31    | 251 |
|  | 8         | 20    | 90     | 60    | 360               | 20    | 271 |
|  | 9         | 30    | 60     | 70    | 361               | 30    | 250 |
|  | 10        | 20    | 80     | 71    | 380               | 40    | 270 |
| Total Time                                       | 260       | 681   | 694    | 3624  | 291               | 2583  |     |
| Array  |           |       | -63%   | -81%  | -11%              | -74%  |     |
| Vector   | 167%      | 432%  |        |       | 138%              | 40%   |     |
| Synch  | 12%       | 279%  | -58%   | -29%  |                   |       |     |

Table 2 Performance comparison

As with Sets, there's a special implementation of Map that supports a List-like view. LinkedHashMap provides for the same deterministic ordering as LinkedHashMapSet and supports all Map operations.

There's another specialized implementation of Map, WeakHashMap, that uses weak references. By employing WeakReference, the garbage collector is able to destroy objects despite the Map's reference. If no other thread holds a reference to a key in the WeakHashMap, the garbage collector is free to collect the key-value pair.

### Abstractions

The framework offers several opportunities for creating your own collection classes. The abstractions are for those instances where you want a more application-specific collection. There are several abstract classes implementing the interfaces with enough basic functionality to make your task less painful (see Figure 5).

In general, you won't be extending these classes unless you want to try some new algorithm or storage technique. Most likely you should turn your attention to the wrapper classes as implemented by the Collections class. Using the Decorator pattern, as these classes do, you may create highly specialized versions of the containers. There's an excellent example in the group of classes created by Piet Jonas for detecting type errors. Using Piet's classes, it's possible to have an exception thrown if an incorrect type is inserted into a collection. These classes employ the exact same design as the specialized wrappers available in the synchronization and read-only methods that I'll discuss next.

### java.util.Collections API

Did you know that many of the Vector's methods are declared with the synchronized modifier? Are you aware of the cost of synchronization? While there have been advancements in many JVMs, there is still a slight overhead incurred with synchronization.

Unless several different threads might access your collection, forget about any of the thread-safe implementations. Use one of the nonthread-safe implementations, like ArrayList or HashMap. If you need index-based access, use the ArrayList. If you are more concerned about key-based access, use the HashMap.

While I may mention Vector and Hashtable from time to time, you should be aware that these two classes are now referred to as legacy code. The API has been reworked of late and all of the collection APIs are now unsynchronized. Special synchronized wrappers have been implemented (and hidden from us) for creating polymorphic, thread-safe implementations of the unsynchronized classes. You gain access to these thread-safe versions via static methods on the Collections class.

```
Collection Collections.synchronizedCollection(Collection);
List Collections.synchronizedList(List);
Map Collections.synchronizedMap(Map);
Set Collections.synchronizedSet(Set);
SortedMap Collections.synchronizedSortedMap(SortedMap);
SortedSet Collections.synchronizedSortedSet(SortedSet);
```

Notice that all of these methods accept the most general interface and return the same interface. If you make judicious use of these generalities, you'll be able to swap out implementations relatively painlessly. Now keep in

mind that in theory, the implementation of collections shall be free to do whatever it wants. You don't want your code dependent upon J2SE source. If you insist on using the concrete classes, you'll have to downcast to use the results from the previous methods. Downcasting requires knowledge of implementation. Things will change over time. Try to insulate yourself from potential change points. The entire Collections Framework wreaks polymorphism, so take advantage of it, as polymorphism is a good thing.

I performed a small test to compare ArrayList, SynchronizedList, and Vector, all three of which implement the List interface. The results show that for synchronized updates, Vector is the worst performer, while SynchronizedList is much faster. Both are compared to the unsynchronized ArrayList. The test involved completing a read (get) or write (add) operation in a tight loop, 100,000 times (see Table 2).

Comparing Vector to SynchronizedList shows that Vector takes 138% and 40% more time than the same operations on SynchronizedList. Meanwhile, SynchronizedList takes a 12% hit over ArrayList for read operations, compared to the 167% increase for Vectors. Some people might be confused by the lack of symmetry in the numbers. If we want to compare A to B, the proper equation is  $(A - B)/B$ . Therefore if I want to compare 2 to 6, then  $(2 - 6)/6$  gives  $-0.6667$  or  $-66\%$ . If I compare 6 to 2, then  $(6 - 2)/2$  gives 2 or 200%. This may seem counterintuitive to saying 6 is three times as large as 2 (which is just a simple ratio, not a comparison).

All of the collections support iterator semantics. Some will bark at you if the underlying collection is altered while you are accessing the iterator by throwing a ConcurrentModification exception.

The static class Collections has many other useful methods for converting to and from certain types of collections. Of interest are those dealing with the creation of unmodifiable collections. First you create your collection and then pass it into the appropriate method and your collection is transformed into something that looks just like the original, but now it will throw an exception if anyone attempts to add or delete an object. Inner classes in Collections that simply extend the standard collection class and override the modifiers accomplish this. Now you can implement the Command pattern and employ the concept of read-only parameters and return structures. In a language that deals exclusively with object references, that's a nice-to-have feature.

```
List Collections.unmodifiableList(List);
Map Collections.unmodifiableMap(Map);
Set Collections.unmodifiableSet(Set);
SortedMap Collections.unmodifiableSortedMap(SortedMap);
SortedSet Collections.unmodifiableSortedSet(SortedSet);
```

Sorting has been taken care of with a "tuned" implementation of Merge Sort. There are routines for sorting primitives and objects. You can implement classes that have a natural order by extending Comparable. If inheritance is at a premium, use the Comparator interface. C++ programmers will feel right at home with this idiom from the STL. There are even collections that are themselves sorted. SortedTree allows you to add objects that will be sorted on the fly. The API is so flexible that you can implement the natural order.

Other utility methods in collections have to do with searching a List. The Collections class offers two binary searching methods.

```
Object Collections.binarySearch(List list, Object key);
Object Collections.binarySearch(List list, Object key, Comparator
comparator);
```

These two methods, one of which employs the natural sort order of the list and the other the ad hoc, run in  $\log(n)$  time where  $n$  is the number of elements in the list. However, this is true only if the list passed in implements the RandomAccess interface. Otherwise, if the list does not implement RandomAccess and is large, the search will execute an iterator-based binary search, which according to the Javadoc will "perform  $O(n)$  link traversals and  $O(\log n)$  element comparisons."

Figure 6 shows the big picture with the preferred extension points highlighted. We've discussed the general categories: Collection (Set, List) and Map. We've played around a little and have seen that to take full advantage of some collections, we have to override Object.equals and Object.hashCode. Also, we went over some of the performance tradeoffs of a couple of implementations. I should mention that there are other Collection APIs available to Java programmers. There's the popular JGL and the JDSL. I haven't looked at the JGL but I have played around with the academic version of the JDSL. The JDSL gives you all those nifty data structures you talked about in your junior year algorithms class.

## Google Seeks Expert Computer Scientists

**Google**, the world leader in large-scale information retrieval, is looking for experienced software engineers with superb design and implementation skills and considerable depth and breadth in the areas of high-performance distributed systems, operating systems, data mining, information retrieval, machine learning, and/or related areas. If you have a proven track record based on cutting-edge research and/or large-scale systems development in these areas, we have plenty of challenging projects for you in Mountain View, Santa Monica and New York.

Are you excited about the idea of writing software to process a significant fraction of the world's information in order to make it easily accessible to a significant fraction of the world's population, using one of the world's largest Linux clusters? If so, see <http://www.google.com/cacm>. EOE.



There are some new collections available in the latest JCF: `LinkedHashSet`, `LinkedHashMap`, and `IdentityHashMap`. In general they are highly specialized versions of the core JCF classes.

## Conclusion

This article should prompt you to take another look at the Collections Framework and, if you are lucky, you'll see something that fits with your current development task. If you are really lucky, perhaps you'll see something else in the J2SE libraries that you never knew existed, collection related or not. Unfortunately, there doesn't seem to be any J2SE champion at Sun, so you'll have to make an effort to scan through the API's Javadoc every so often and perhaps even the source code as well (there are some novel snippets in there). ☺

## References

- *The Collections Framework*: <http://java.sun.com/j2se/1.4.2/docs/guide/collections/index.html>
- *Java Collections API Design FAQ*: <http://java.sun.com/j2se/1.4.2/docs/guide/collections/designfaq.html>

- *Annotated Outline of Collections Framework*: <http://java.sun.com/j2se/1.4.2/docs/guide/collections/reference.html>
- *Collections Framework Tutorial*: <http://java.sun.com/docs/books/tutorial/collections/index.html>
- *Introduction to the Collections Framework Short Course*: <http://developer.java.sun.com/developer/onlineTraining/collections/Collection.html>
- Jonas, P. "Secure Type-safe Collections": [www.javaworld.com/javaworld/jw-04-2001/jw-0427-collections.html](http://www.javaworld.com/javaworld/jw-04-2001/jw-0427-collections.html)
- Goetz, B. "Threading lightly, Part 1: Synchronization is not the enemy": [www-106.ibm.com/developerworks/java/library/j-threads1.html](http://www-106.ibm.com/developerworks/java/library/j-threads1.html)
- Eck, D.J. *Programming with Collections, Introduction to Programming Using Java*: Chapter 12 "Generic Programming and Collection Classes," version 4.0, July 2002. <http://math.hws.edu/javanotes/c12/index.html>
- Bell, J. "It's a Modern Day Cinderella." *Java Developer's Journal*, Vol. 8, issue 9.
- *JDSL*: [www.cs.brown.edu/cgc/jdsl/](http://www.cs.brown.edu/cgc/jdsl/)
- *JGL*: [www.recursionsw.com/products/jgl/jgl.asp](http://www.recursionsw.com/products/jgl/jgl.asp)

### Listing 1: Person Class

```

1 // Person Class
2 public class Person {
3     private String ssn;
4
5     public Person(String newSSN) {
6         this.ssn = newSSN;
7     }
8
9     public String toString() {
10        return ssn;
11    }
12 }
```

### Listing 2: PedanticPerson

```

1 // PedanticPerson Class overrides
2 // hashCode and equals.
3 public class PedanticPerson
4     extends Person {
5
6     public PedanticPerson(
7         String newSSN) {
8         super(newSSN);
9     }
10
11    public int hashCode() {
12        return getSSN().hashCode();
13    }
14
15    public boolean equals(
16        Object obj) {
17        Person p = (Person) obj;
18        return getSSN().equals(
19            p.getSSN());
20    }
21 }
```

### Listing 3: Main Program

```

1 import java.util.HashSet;
2 import java.util.Set;
3
4 public class HashSetExample {
5
6     public static void main(
7         String[] args) {
```

```

8
9     Set dupSet = new HashSet();
10    dupSet.add(new
11        Person("000-11-1111"));
12    dupSet.add(new
13        Person("222-23-1234"));
14    dupSet.add(new
15        Person("000-11-1111"));
16
17    // has duplicates
18    System.out.println(dupSet);
19
20    Set set = new HashSet();
21    set.add(new
22        PedanticPerson("000-11-1111"));
23    set.add(new
24        PedanticPerson("222-23-1234"));
25    set.add(new
26        PedanticPerson("000-11-1111"));
27
28    // has no duplicates
29    System.out.println(set);
30 }
31 }
```

### Listing 4: PersonComparator

```

1 // PersonComparator will be used
2 // to sort the Persons in
3 // descending order based
4 // on SSN.
5 public class PersonComparator
6     implements Comparator {
7
8     public int compare(
9         Object o1,
10        Object o2) {
11
12        Person p1 = (Person) o1;
13        Person p2 = (Person) o2;
14
15        String ssn1 = p1.getSSN();
16        String ssn2 = p2.getSSN();
17
18        return ssn2.compareTo(ssn1);
19    }
20 }
```





©2003 Wily Technology, Inc. The Wily logo is a trademark of Wily Technology, Inc. Java is a trademark of Sun Microsystems in the U.S. and other countries.

Two years without a vacation.  
The application's up. It's down.  
It's up. It's down.

I'm to blame. Steve's to blame.  
Someone's always to blame.

Not any more.

Get Wily.™

Enterprise Java  
Application Management  
1 888 GET WILLY  
www.wilytech.com

**wily**  
technology



# Java New Input/ Output

Incorporate NIO functionality  
in your applications

by Vishwanath K

**I**nput/Output (I/O) is one of the fundamental aspects of computing that you have to deal with at some point during the application's development phase. Dealing with I/O presents its own challenges because I/O access is still slow. For example, reading or writing data from or to a disk involves a choreography of electronic, mechanical, and computer engineering disciplines, making the task slow in comparison to, say, reading or writing from random access memory over a high-speed bus.

To minimize the impact of working with slow I/O mediums and to maximize throughput and performance, software designers have devised a variety of strategies. These strategies include, but are not limited to, vectored I/O (scatter/gather) and multiplexing I/O. These strategies can be found in use in I/O-intensive applications, typically written in C or C++. However, in the world of Java such strategies were not applicable and a developer who wants to write a scalable, high-performance I/O-intensive application would, most likely, resort to writing native code.

This situation drastically changed with the introduction of the New Input/Output (NIO) packages along with the Merlin release (JDK 1.4). The NIO packages introduced an array of new functionalities, including improved performance in Buffer Management, Scalable Network & File I/O, Character-Set support, and Regular Expression Matching. All of these enable a developer to write portable, high-performance, and scalable I/O-intensive applications.

This article provides an overview for developers who are planning to incorporate the NIO functionality in their application and for developers who are looking for technologies that enable them to write high-performance I/O applications. This article provides an introduction to the buffers, channels, memory-mapped files, file locking, and multiplexing I/O.

## High-Performance Input/Output

With the current I/O architecture (JDK versions prior to 1.4), developers used streams to perform I/O operations. The basic sets of streams that make up the bulk of the I/O are byte streams and character streams. These streams provide APIs to

carry out fundamental operations, such as reading data from a stream into an array and writing data from an array into a stream. The stream-based I/O blocked while the core I/O operations took place. In other words, a read() method does not return until all the data is read from the stream and, similarly, a write method does not return until all the data is written to the stream. This blocking nature of I/O resulted in performance bottlenecks while writing high-performance I/O-intensive applications.

Apart from blocking, working on raw bytes as they become available is a cumbersome process. Buffering the data as a method of improving the performance was not incorporated in the core design of streams. This means that every single byte written to a stream is passed on to the operating system and then flushed to the physical or the network medium. This is a very inefficient way of handling I/O and results in performance issues.

A common strategy was to explicitly insert a buffering class following the decorator design pattern. This mechanism is incorporated in the core I/O API by means of buffered streams such as `BufferedReader`. To use buffered streams you have to pipe a nonbuffered stream into a buffered stream and then work with it. Working with buffered streams certainly alleviates the issue of working with raw bytes, but then the buffering logic was hidden deep in the bowels of the buffered streams. This strategy provided the application developer with little or no control over the buffers. Also, using these buffers is not very efficient because it involves a lot of data copying from user space buffers to O/S buffers via the JVM buffer.

The NIO framework introduced with JDK version 1.4 addressed all the shortcomings of stream (and buffered stream) based I/O and also provided a suite of new functionalities. To understand the fundamental concept of NIO, there are two aspects that need to be mastered: buffers and channels. In the following sections, we will look at buffers and channels in detail.

## Buffers

A buffer is a container that can hold a finite and contiguous sequence of primitive data types. It's essentially an object wrapper around an array of bytes with imposed limits. A container that is 180 degrees apart from the buffer is an `ArrayList`, which, in theory, is capable of holding an unlimited amount of data. The buffers were introduced not only to provide the application developer with more control, but also to speed up the I/O application.



**Vishwanath K**, PMP, is a project lead at Cap Gemini Ernst & Young in Overland Park, KS. He has about eight years of IT experience spanning a variety of technologies.

[viswanath.krishnan@cgey.com](mailto:viswanath.krishnan@cgey.com)

The buffer is implemented as an abstract class in the java.nio package and has seven direct descendants: ByteBuffer, CharBuffer, DoubleBuffer, FloatBuffer, IntBuffer, LongBuffer, and ShortBuffer. A buffer is characterized by three important properties: capacity, limit, and position. These properties are set during buffer creation and during buffer manipulation. A buffer is created in two possible ways – either by calling the allocateDirect() factory method or by calling the allocate() factory method. Both of these methods take an int as a parameter, which represents the newly created buffer's capacity. The difference between the two buffer creation strategies is that the allocateDirect() method creates a native buffer that is outside the JVM heap. This circumvents the extra copying that's normally required between the JVM buffer and O/S buffer, resulting in a marked improvement in performance. However, this improvement comes at a price – there's a higher cost associated with the setup and teardown of direct buffers. But with cunning programming practices, such as buffer creation during application startup, the cost of creation/destruction could be balanced down in favor of marked improvement in performance.

```
ByteBuffer buffer = ByteBuffer.allocate(512);
ByteBuffer directBuffer = ByteBuffer.allocateDirect(512);
String str = "Hello";
byte[] data = str.getBytes();
buffer.put(data);
```

The first two lines of this code snippet result in the creation of a nondirect ByteBuffer and a direct ByteBuffer. The newly created buffer will have the properties indicated in Figure 1. Data is added to this newly created buffer by calling any of the overloaded put() methods with appropriate parameter(s). In the code example we add the string "Hello" to the byte buffer. After calling put(data) the buffer now has data. The properties of the buffer are altered as indicated in Figure 1.

As we can see, the position pointer now points to the next empty cell after the data. So if we are to use this buffer to read the data and possibly make some business sense out of it, we need to flip the position of the position pointer. This is accomplished with the following code snippet.

```
buffer.flip();
int limit = buffer.limit();
byte[] data = new byte[limit];
buffer.get(data);
System.out.println(new String(data));
```

The flip() method readies the buffer for draining by resetting the position and limit pointer. The capacity pointer is left unchanged. After the flip() is called, the position pointer points to the first cell, and the limit pointer points to the cell where the position pointer used to point before the flip() method was called. A clarification on the difference between limit and capacity is in order at this time. Capacity is the maximum number of items a buffer can hold, whereas limit is a value that ranges from zero to capacity, representing an arbitrary limitation for the buffer. The limit is set by calling either the limit() method or the flip() method.

It should be noted that the bytes are ordered in a certain fashion, which is based on the byte ordering supported by the platform. For example, byte ordering in x86 architecture is little-endian and the byte ordering in the Unix platform is big-endian. However, the byte ordering can be explicitly set by the order(ByteOrder byteOrder) method on the buffer. The order()

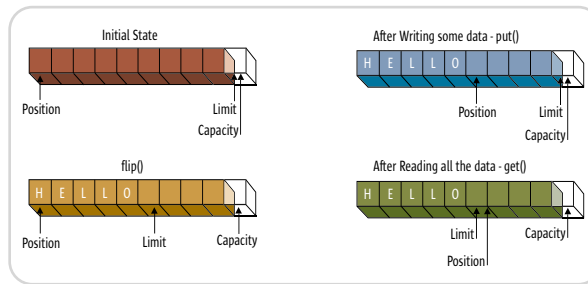


Figure 1 Buffer properties

method (without parameters) returns the current byte ordering for this buffer. Figure 2 demonstrates the difference in little-endian and big-endian byte ordering of the magic number of the Java byte code.

## Channels

A channel represents an open connection to an entity such as a hardware device, a file, a network socket, or a program component that is capable of performing one or more I/O operations. For example, we could have a SocketChannel or a FileChannel that represents a connection to a socket or to a file, respectively. The main difference between streams and channels is that for the former we need to have separate streams to do reading and writing, but in the latter case we need only one channel to accomplish both reading and writing. The other difference is that streams operate on raw bytes, whereas channels operate on buffers.

The Channel interface and its extension are defined in the java.nio.channels package.



### Build Incredible Interactive Diagrams

#### Feature Packed New JGo Release!

JGo, written entirely in Java, makes it easy for your Java applications to create interactive diagrams of all kinds. Create network editors, schematics, process flow diagrams, visual languages, organization charts, family trees, etc.

The JGo class library provides containers, connectors, links, orthogonal routing with automatic node avoidance and enhanced link options, drag-and-drop, scaling, in-place text editing, tooltips, printing, SVG/XML, nested graphs, lots of new sample nodes and applications, a high performance Auto Layout Option, improved IDE integration, and more...

- **FREE Evaluation Kit**
- **No Runtime Fees**
- **Full Source Code**
- **Easy to Learn & Use**
- **Excellent Support**



**www.nwoods.com/go/**  
**800-434-9820**  
*Ask us about SWT!*





Figure 3 provides a high-level view of the Channel interface and its subinterfaces.

The three interfaces – ByteChannel, ReadableByteChannel, and WritableByteChannel – are fairly self-explanatory about their functions. The ScatteringByteChannel and GatheringByteChannel provide the means for reading or writing a sequence of bytes from multiple buffers in a single invocation. Scattering and gathering, also known as vectored I/O, have been around for a while and are widely used for developing high-performance I/O applications. A real-world example that makes extensive use of this concept at the device level is an SCSI controller. Listing 1 demonstrates the application of ScatteringByteChannel in a conjured-up application – a WAV player that plays WAV files. A WAV file is made up of three major components: the RIFF chunk (12 bytes) that identifies the file as a WAV file; the format chunk (24 bytes) that identifies parameters such as sampling rate, channels, bytes/second, etc.; and the data chunk (the rest) that contains the actual data in bytes. Listing 1 shows how to read all this data in a single invocation.

Scattering, using an array of ByteBuffer, results in a data transfer in a single method invocation. This technique avoids the need for multiple system calls to perform the reads, and combines all reads into one optimized read system call. The result – a performance boost through the means of optimized data transfers to/from variable-size buffers. Similar logic holds true for GatheringByteChannel.

#### SelectableChannel

A SelectableChannel is a channel that can be multiplexed by a Selector. The details about the workings of the Selector are elaborated on later in the article. The descendants of SelectableChannel are DatagramChannel, Pipe.SinkChannel, Pipe.SourceChannel, ServerSocketChannel, and SocketChannel. In this section we'll quickly glance through the purpose of a few popular channels.

#### ServerSocketChannel and SocketChannel

ServerSocketChannel is a selectable channel for stream-oriented sockets (ServerSocket). This creates a socket that accepts the inbound client connections. This socket cannot read or write. Binding and socket manipulation must be done by using the channel's peer – the ServerSocket, which can be obtained by the socket() method. SocketChannel is a selec-

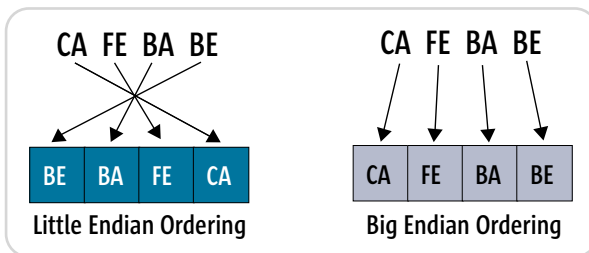


Figure 2 Byte ordering

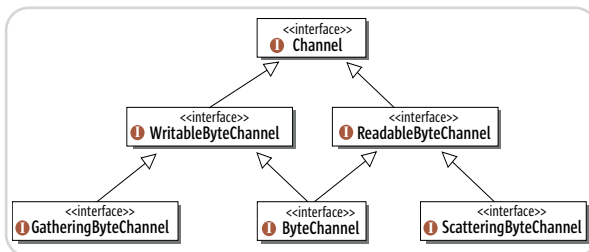


Figure 3 NIO class hierarchy

table channel for stream-oriented sockets (Socket). This is an abstraction of the Socket with the added functionality of non-blocking operations. Binding and socket manipulation must be done using the channel's peer – Socket, which can be obtained by the socket() method.

#### FileChannel

FileChannel is probably the most frequently used channel, as it's used for reading, writing, mapping, and manipulating the file. With FileChannel it's now possible to do more than just reading or writing. Two of the major functionalities that could be accomplished using FileChannel are discussed below.

#### Memory Mapped Files

Memory mapping of a file involves mapping certain portions of the file or the entire file directly into memory. Any changes that are made to the mapped regions of the file are flushed to the underlying file. This is a piece of functionality that was missing in Java but that could be done using C or C++ starting from MUTICS O/S. These procedural languages use the mmap() function to map a file into the memory. The resulting file pointer is then used to manipulate the file. In Java, with NIO it's now possible to map an entire file or portions of a file into memory using the following code.

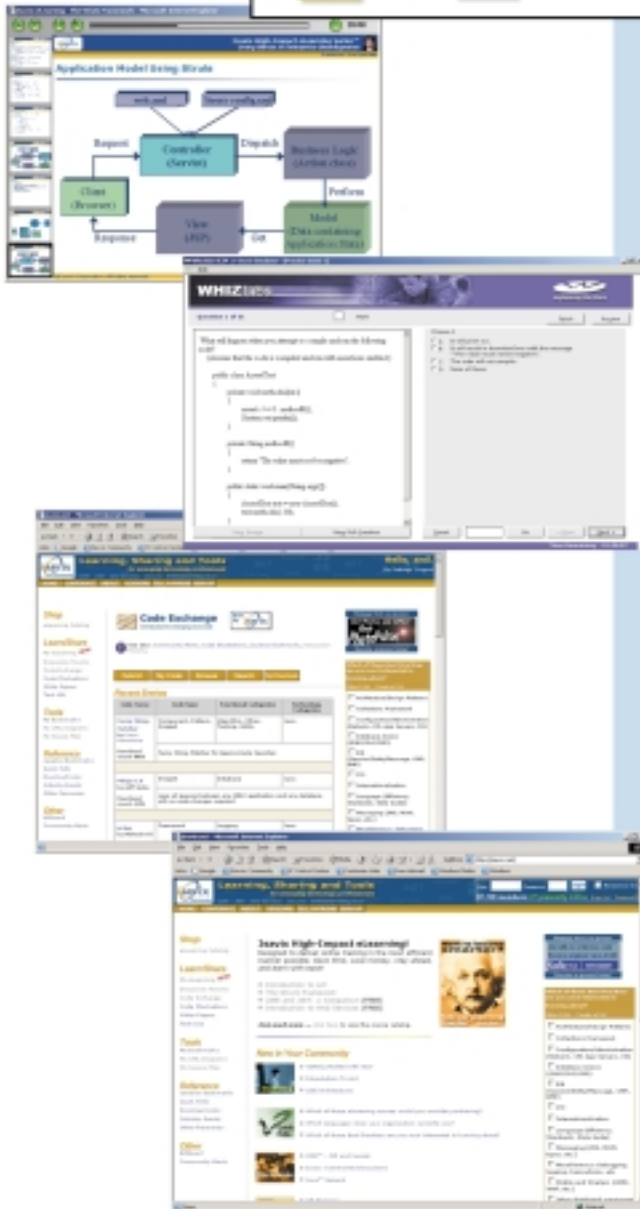
```
File f = new File("LargeMemoryMapFile.dat");
RandomAccessFile raf = new RandomAccessFile(f, "rw");
FileChannel fileChannel = raf.getChannel();
MappedByteBuffer mappedByteBuffer =
fileChannel.map(MapMode.READ_WRITE, 0, f.length());
```

The map() method of the FileChannel returns a MappedByteBuffer, a particular kind of direct buffer. The main advantage of mapping a file is that the file that's mapped into memory results in the O/S mapping the file as its virtual memory. If a particular portion of the file is modified, only that part is brought into physical memory by means of a page fault. This is particularly advantageous while working with files that are large in size (in order of gigabytes).

As an example, if you were to write a text editor for a Win32-bit system, then you could potentially work with files larger than two gigabytes. Working with such large files could be a major drain on performance unless it's done using memory-mapped I/O. However, when dealing with small files that are on the order of a few kilobytes, memory mapping doesn't translate into any performance advantage.

It is possible to load the contents of the entire file into physical memory using the load() method of the MappedByteBuffer. Mapping the entire file into physical memory using load() will result in a large number of page faults (at the O/S level), which will be a serious drain on the application performance. This is especially true while working with large files. The solution to this problem is to map only certain portions of the file into memory. This is accomplished by providing the appropriate position and size as a parameter for the map() method of FileChannel.

In addition, the mapping could be done in any of the three possible modes: Read Only, Read/Write, and Private. In Read Only mode only the read operation could be done on the buffer. Any attempt to write to the buffer will result in a checked exception being thrown. In Read/Write mode it's possible to carry out both read and write on the buffer. In Private mode, any changes made to the buffer will not be propagated to the file and the changes will not be visible to other programs that might have mapped the same file.



## Searchable Books

- Conduct searches across 1500+ technical books from publishers such as Addison Wesley, Microsoft Press, O'Reilly, Prentice Hall, and many more. Zero in on answers to time-critical questions instantly.
- Read the books on your Bookshelf from cover to cover. Or, simply flip to the page you need.
- Browse books by category. Researching any topic is a snap. From XML, to database to .Net, you'll find your answer here.

## eLearning Courses

The Isavix High-Impact eLearning Series™ is designed to deliver online training in the most efficient manner possible. Each course is approximately one hour in length with no "homework" assignments and can be viewed in Flash and PDF formats. Sample courses include The Struts Framework, Introduction to Apache ANT, ASP.NET Crash Course, and more.

## Certification Exam Preparation

Whizlabs exam simulators are available for Sun, IBM, BEA, CISCO, Microsoft, Oracle, CompTIA, CIW and Red Hat certifications. These products have been developed by experts who are considered the authorities in their subjects.

From computer professionals to Certification Training centers to College and University students, to hundreds of corporations ... the Whizlabs solutions have helped over 150,000 people get their certifications.

## FREE Resources and Tools

isavix.net is a site that provides eLearning, information sharing, software exchange, helpful tools, and valuable links for emerging technology (e.g. J2EE, .NET) professionals.

Our free tools and resources include Discussion Forums, Code Exchange, My Secure Files, Quick Polls, My Bookmarks, Download Links, My UML Diagrams, My eLearning, and much more.

## Custom Community Portal For Your Organization

If you like the many benefits isavix.net provides but would want a custom version of our portal available exclusively for your organization, we can do that for you!

Our community can either be installed within your organization's Intranet or provided to you as an ASP solution.

The benefits include an enhanced user experience, professional development, improved team dynamics, effective project management, and better resource allocation. The ROI for an organization is high because developers can save from a few minutes to hours a day since the relevant information is readily available to them, when they need it!

**Corporate and Government solutions  
available ... contact us for details.**



**Isavix Corporation**  
Herndon, VA  
1-866-472-8849  
info@isavix.com

## File Locking

With the current I/O, the ability to lock a file is not available as part of the API. To implement this functionality a developer has to write JNI code, thereby making the code nonportable (defeating WORA). The NIO introduced the ability to lock the file, providing clean, consistent, and 100% portable code. File locking is typically required when there is a need for data sharing and mutual exclusion among applications. The file locks are built right into FileChannel and it's now possible to lock files on any O/S that supports the file-locking functionality. This is achieved with the code in Listing 2.

The lock() method acquires an exclusive lock on the file channel. It's also possible to obtain a lock on a section of a file as opposed to locking the entire file. This is possible through the call lock method (long position, long size, boolean shared).

A file lock is held until the release() method is explicitly called or until the file channel is closed. A word of caution: certain O/Ss don't permit a locked file to be mapped into memory and vice versa. Therefore, programs that rely on memory mapping and file locking may not be portable.

## Selector

There are times when a developer has to wrestle with handling I/O from multiple data sources. With traditional, blocking stream-based I/O, the typical strategy is to spawn multiple threads to read/write data from multiple streams. This is done because by using a single thread to read/write data from/to multiple streams, we run the risk of blocking when there's no data available in the stream. However, spawning multiple threads is not a target state solution because of the limitation it introduces. A thread when created is allocated up to one

megabyte of contiguous memory. In a 32-bit process, we're limited to a maximum of four gigabytes of memory (2<sup>32</sup> bytes) – meaning we'll start getting java.lang.OutOfMemoryError before we hit around 4,000 threads. Although 4,000 threads may sound like too many, for a serious multithreaded high-performance application, 4,000 threads is just not enough (think of a multithreaded Web server).

When handling data from multiple data sources, a procedural language such as C doesn't suffer from Java's limitations. Handling multiple data sources in C is achieved by means of a select() system call. However, with the release of Merlin Java (version 1.4), the inability to deal with multiple data sources in Java was finally put to rest. This release injected new life into the world of scalable I/O by means of multiplexing I/O. Multiplexing I/O becomes possible when a read/write operation doesn't block.

Before we look at nonblocking I/O in detail, let's take a quick peek at traditional blocking I/O. In a stream-based application, when an I/O operation is performed, say a read() operation, the method is blocked until some data is made available.

Figure 4 demonstrates the steps involved in the data transfer between two systems using network sockets.

The data transfer between a client and the server involves multiple buffer copying and transfers through the network. So when the client enters the read() method, the entire data may not be available in the buffer. The client blocks until the entire requested data is made available to the client. In a nonblocking I/O, the read call returns immediately with whatever data is available. However, working with incomplete data does not serve a purpose and therefore the client has to resort to polling, which involves sitting in a tight loop waiting for all the data to become available.

Polling results in burning CPU cycles and is, therefore, considered inefficient. A better mode of operation would be an event-driven mechanism where an appropriate notification takes place when the data becomes available. This is what the Selector mechanism provides. This notification mechanism, coupled with nonblocking functionality, enables developers to write high-performance, scalable I/O-intensive applications.

A note on the Reactor design pattern is in order at this point. The Reactor pattern decouples the events arrivals from event handling. The events arrive at an arbitrary time and are not dispatched immediately. The reactor keeps track of the arrived events and dispatches only when the handler asks for them. This architecture is indicated in Figure 5.

The Selector class that's found in the java.nio.channels package plays the role of the "Reactor" as stated in the Reactor design pattern. The Selector multiplexes events on the SelectableChannel. The channels that extend the SelectableChannel class could be placed into a nonblocking mode using the configureBlocking() method call. This means that the three channels – ServerSocketChannel, SocketChannel, and DatagramChannel – that extend AbstractSelectableChannel could be placed in a nonblocking mode. However, FileChannel cannot be placed in a non-blocking mode.

A reference to the Selector is obtained through the static method open() on the Selector class. A reference to ServerSocketChannel or SocketChannel is obtained through the static method open() on those classes. The ServerSocketChannel and SocketChannel could be placed in nonblocking

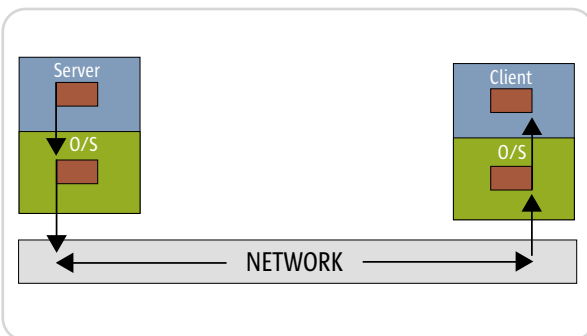


Figure 4 Data transfer across the network

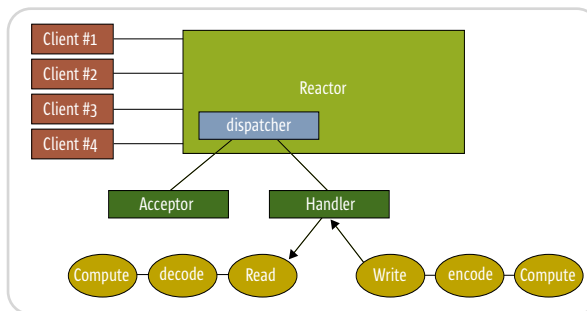


Figure 5 Reactor

| Class               | Valid Operation               |
|---------------------|-------------------------------|
| ServerSocketChannel | OP_ACCEPT                     |
| SocketChannel       | OP_CONNECT, OP_READ, OP_WRITE |
| DatagramChannel     | OP_READ, OP_WRITE             |

Table 1 Valid operations



live wireless.

work wireless.

be wireless.



The most important  
technology event  
of the year!

## CTIA WIRELESS 2004

is the one show where wireless standards are created and the technological direction of the industry is set. With the largest gathering of wireless engineers and technologists, this is where you will find the tools you need to help build and advance the wireless industry.

**Look at all CTIA WIRELESS 2004 has to offer the wireless engineer and technologist:**

- 6 CTIA educational sessions dedicated to exploring wireless technology
- IEEE Wireless Communications Network Conference (WCNC) 2004 – the industry's foremost conference for developing wireless standards and engineering
- WiFi Summit – the CTIA Smart Pass program, a cutting edge look at WiFi strategy and security
- A 400,000 square foot exhibit floor displaying the latest in wireless technology and applications



welcome the wireless generation.

March 22-24, 2004

Georgia World Congress Center

Atlanta, GA, USA

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

John T. Chambers  
President & CEO  
Cisco Systems



Scott McNealy  
Chairman & CEO  
Sun Microsystems, Inc.



Russell Simmons  
Chairman & CEO, Rush Communications,  
Co-founder & Chairman, Def Jam Records



mode by simply invoking `configureBlocking()` with `false` as the method parameter. This is demonstrated in the following code snippet.

```
Selector selector = Selector.open();
SocketChannel channel = SocketChannel.open();
channel.configureBlocking(false);
channel.register(selector, SelectionKey.OP_CONNECT |
SelectionKey.OP_READ);
```

The next step is to register the channel with the selector (not the other way around), indicating the events of interest. This is done using the `register()` method of the `SelectableChannel` abstract class. The event of interest depends on the type of selectable channel. Table 1 presents valid operations for a few subclasses of the `AbstractSelectableChannel`.

After registering the channel with the selector, the next step is to wait till the Selector indicates that an event of interest has occurred. This is achieved through the `select()` method, which blocks until an event of interest occurs. When an event of interest does occur, the `select()` method returns with an integer indicating the number of updated keys. A key is an instance of the `SelectionKey` class that defines the relationship between the `SelectableChannel` and the `Selector`. As an aside, a selection key is created each time a channel is registered with the selector. The key remains valid until it is cancelled by invoking its `cancel()` method on the `SelectionKey`. The selector maintains three different sets of selection keys:

- The key set that represents the set of channels that are registered with the selector
- The selected key set that represents the set of channels that are detected to be ready for at least one of the operations identified in the key's interest set during a prior selection operation
- The cancelled key set that represents the set of keys that has been cancelled but whose channels have not yet been deregistered

When the `select()` method returns the updated set, we need to look for the selected key set that's obtained by the `selectedKeys()` method. Once we have the selected key list, we iterate through the list of keys. For each key we get the corresponding channel that generated the event of interest by invoking the `channel()` method. In the case of the server, this method returns `ServerSocketChannel`, which could be used to accept the incoming client connection. With the `SocketChannel` you need to check which event might have occurred. The previous code snippet indicated that we have expressed interest in `connect` and `read`. The following logic can be used to determine which event caused the `select` to return.

```
while (selector.select() > 0) {
Set keys = selector.selectedKeys();
Iterator readyIter = keys.iterator();
while (readyIter.hasNext()) {
SelectionKey key = (SelectionKey) readyIter.next();
if (key.isConnectable()) {
// Do something...
} else if (key.isReadable()) {
// Do something...
}
}
}
```

Keep in mind that while iterating over the selected key set, it's imperative that we remove the key that was just obtained.

This is necessary because the selected key set might have been updated while the processing was going on. This is achieved with the following code:

```
while (readyIter.hasNext()) {
// Get key from set
SelectionKey key = (SelectionKey)readyIter.next();
// Remove current entry
readyIter.remove();
...
}
```

Thus using the `Selector` class in conjunction with threads enables a developer to write a high-performance, non-blocked, and scalable I/O application. An example reactor implementation is provided along with this article, which details the concepts described above. The source code can be downloaded from [www.sys-con.com/java/sourcec.cfm](http://www.sys-con.com/java/sourcec.cfm).

## Conclusion

The new I/O features introduced in the Java version 1.4 release provide exciting new ways to improve the performance and scalability of your I/O-intensive applications. This is especially good news for server-side developers, who can now use this powerful feature to develop robust applications without resorting to nonstandard coding practices. ☺

## References

- *New I/O APIs documentation*: <http://java.sun.com/j2se/1.4.1/docs/guide/nio/index.html>
- Tanenbaum, A. (2001). *Modern Operating Systems, Second Edition*. Prentice Hall.
- Hitchens, R. (2002). "Top Ten New Things You Can Do with NIO." O'Reilly Network: [www.onjava.com/pub/a/onjava/2002/10/02/javanio.html](http://www.onjava.com/pub/a/onjava/2002/10/02/javanio.html)

### Listing 1

```
File file = new File(fileName);
RandomAccessFile raf = new RandomAccessFile(file, "r");
FileChannel fileChannel = raf.getChannel();

ByteBuffer riffChunk = ByteBuffer.allocate(12);
ByteBuffer headerChunk = ByteBuffer.allocate(24);
ByteBuffer dataChunk =
ByteBuffer.allocate((int)file.length()-12-24);

ByteBuffer[] wavChunks = {
    riffChunk, headerChunk, dataChunk
};

fileChannel.read(wavChunks);
```

### Listing 2

```
File f = new File("FileToBeLocked.dat");
RandomAccessFile raf = new RandomAccessFile(f, "rw");
FileChannel fileChannel = raf.getChannel();
FileLock fileLock = fileChannel.lock();
if (fileLock.isValid())
    doSomethingHere();

fileLock.release();
// or close the channel releases the lock!
```



# Forget something?

**Post-launch is NOT the time to be verifying web applications.**

**The wild blue yonder of operational monitoring and management is extremely unforgiving.** Which means that going live with the monitoring software you used in development is a great way to go dead—quickly! You simply can't support operations if your staff is drowning in details provided by development profiling tools and debuggers. **Let NetIQ cover your apps...with AppManager.**

AppManager—the industry's easiest-to-use Systems Management suite—is a proven management system for monitoring J2EE application servers, databases, operating systems and even end-user response time. NetIQ's AppManager monitors ALL application components—not just your server. **NetIQ. Nobody does UNIX better. Nobody.**

Visit us at [www.netiq.com/solutions/web](http://www.netiq.com/solutions/web) to learn how we can help you address the challenges of your operational monitoring and management.





# DevPartner 3.0.1 Java Edition

by Compuware Corporation

Reviewed by  
Vijay Phagura

**D**evPartner Java Edition is a profiling tool from Compuware that helps developers envision the reality of their designs and implementations. It clearly shows the performance, memory, and code coverage of various modules in your project. If you always wondered how your particular implementation would behave and want to buy a tool to show the profiling statistics of your project, this review should interest you.

DevPartner is a good tool for getting a correct picture of your application. It tells you how it is performing, how much memory each module of the application takes when executing, and how many times a piece of code has been executed.

It can be downloaded, installed, and configured very easily; I didn't run into any problems. DevPartner can be attached to various application servers for profiling. It does cover the popular servers like WebSphere, WebLogic, etc., but I wish it could be hooked to others as well, for example, JBoss. Figure 1 shows one such setup screen for Apache's Tomcat server.

The server is automatically run by DevPartner when the user hits the Continue button. The state of the server is also indicated on the upper panel of this screen. As the server is executing an application, profiling data can be collected and viewed at any time.

The three main features – Performance, Memory Analysis, and Code Coverage – can be selected by a dropdown from the configuration screen shown in Figure 1. To change the feature from Performance to Memory Analysis, first stop the server, select Memory Analysis from the dropdown, then restart the server. This clearly is a drawback and a tedious process.

## Performance

This tool measures performance fairly accurately in many different ways, such as percentages, average time, etc. To see problematic areas at a glance in an application, the percentage method is quick to show the user the areas where more time is being spent, and it can show you how much time is spent in each method of a class. The user can also browse the source code, if the paths are set right.

## Memory Analysis

DevPartner also shows the memory used by any part of an application. The representation is very graphical and the user is allowed to drill down to methods and also browse the source code. It can show memory on a per-thread basis too.

Memory analysis is also shown in real time as the project is executed by the server. The user can choose to run garbage collection manually by hitting a button on a screen (see Figure 2).

## Compuware Corporation

One Campus Martius  
Detroit, Michigan 48226  
**Phone:** 800 521-9353  
**Web:** [www.compuware.com](http://www.compuware.com)

## Specifications

**Platforms:** Windows XP/2000/NT/98/ME; Solaris 8 and 9; Red Hat Linux 7.3 or 8.0 Personal, Professional, and Advanced Server Editions

**Java Virtual Machine:** Sun 1.3.1 (32-bit as Classic and Hotspot) or 1.4 (32-bit); IBM 1.3.1 (32-bit as Classic and Hotspot) or 1.4

**Java Application Servers:** Windows, Solaris and Linux; BEA WebLogic 6.1, 7.0; IBM WebSphere 4.0.3 and 5.0, Advanced, Advanced Single Server, and Advanced Developer Editions (also available on AIX); Oracle9iAS v9.0.2 or v9.0.3, Standard and Enterprise Editions; Sun ONE Application Server (formerly known as iPlanet) 6.5 and 7.0; Tomcat 3.3.1 or 4.1

## Test Environment

Win XP Pro, 2GHz CPU, 512MB RAM



Figure 1 Setup screen for Apache's Tomcat server

## Code coverage

Yet another presentation is the code coverage. Here the tool gives you a count of the number of times a part of the application has been executed during a particular run. This can be very helpful to efficiently trimming the code to improve performance.

## Other Features

With each of these features DevPartner provides useful views. One such example is the method call trace view (see Figure 3). This gives a pictorial view of how various methods are called in an application during execution.

Vijay Phagura, a professional Java/J2EE consultant, has over 12 years of experience in software architecture and development. He specializes in designing and developing software using J2EE and other Java technologies.

[vphagura@yahoo.com](mailto:vphagura@yahoo.com)



Figure 2 The user can run garbage collection manually

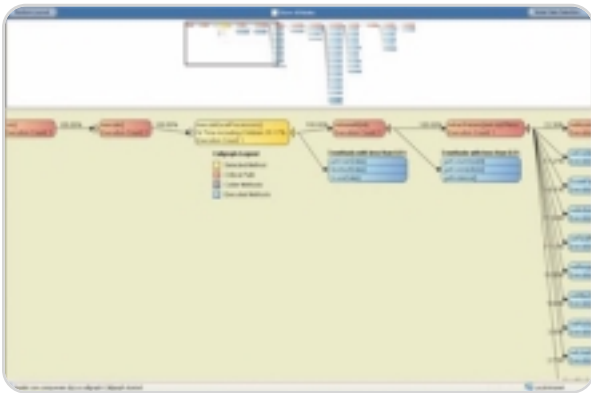


Figure 3 The method call trace view

## Snapshot

**Target Audience:** Software developers

**Level:** Beginner to advanced

**Pros:**

- Easy to use
- Configurable
- Descriptive views
- Source code browser

**Con:**

- Supports limited application servers

Another feature worth noting – it can be integrated with an IDE. Unfortunately, the IDEs that it supports are limited. It supports JBuilder and other application server–embedded IDEs.

When you start using DevPartner you'll notice that you don't have to go to its help that often, as each button provides help statements.

## Summary

I used DevPartner on a project with a Web application running on Tomcat 4.1.18. It performed fairly well compared to similar tools in the market. It was fast to start and restart. Also, it was pretty responsive when viewing the source code and gathering statistics for a run.

Overall, I found DevPartner to be very useful and it can save you tons of time and effort while fine-tuning an application. I really would like it if it supported more application servers and IDEs, as this tool can increase the quality of your applications. ☺

# WebCharts3D

## One demo worth your download

[www.webcharts3d.com/demo](http://www.webcharts3d.com/demo)



WebCharts 3D is the ideal XML-based charting solution for web applications.

With its powerful charting engine, WebCharts 3D can produce a wide range of different charts in the most demanding enterprise environments.

Deliver charts to browsers and mobile devices as either applets or interactive server-generated images in a variety of popular formats. Or, embed JavaBeans directly into rich client applications.

WebCharts 3D's WYSIWYG approach allows you to begin development immediately after installation.



**Fastest Development.  
Fastest Deployment.  
Fastest Runtime.**

# Behind the Glass



**Joe Winchester**

Desktop Java Editor



**R**ecently I was giving a demo of Java Web Start (JWS) to a customer and while they appreciated that systems management issues had been addressed, someone in the audience said “it’s just client/client all over again – not really client/server.” Her point was that true client/server is about the runtime separation of the two environments, not just deployment magic. It was my fault because I had used a “Hello World” program to demonstrate JWS, first running it locally and then deploying it across JWS in a “look no hands, ma” kind of fashion. My demo slide even included the bullet point that no changes were required in the Swing program to allow it to run across JWS. She told me, in nice words, that I didn’t get it because changes should be required. The runtime dynamics of how the program should validate input and access data are different when run locally or across a client/server divide.

What she was looking for was a mechanism for building a Swing program that had several pieces to it: a client that concerned itself with presentation, local validation, and a richer UI experience than HTML currently offered; a server that dealt with issues such as persistence, concurrency, and load balancing; a bit in the middle that was the runtime transport layer; and a distribution mechanism for which JWS appeared to be satisfactory.

levels of accessibility, countries, or hardware devices could be achieved by providing new XML files that were dispatched according to each client type.

## Use Web Services Between the Client and Server

Communication between the two sides could be done using Web services, so the server components are managed as a pool of shared resources in a J2EE server that each client communicates to. Having a defined protocol between the two portions of the application insulates each side, so new implementations of existing services can be swapped in without disrupting the client, and likewise new clients can be created combining multiple services for a single functional task. Unlike JDBC, where the program’s SQL exposes the structure of the server and is thereby vulnerable to structural changes, Web services tend to be more task-oriented, like a more traditional transaction API providing both physical and logical separation.

For the GUI portion the beginnings of this might already be there with technologies such as SwixML ([www.swixml.org](http://www.swixml.org)), SwingML (<http://swingml.sourceforge.net/>), or the XML-Encoder/XMLDecoder that was included with 1.4. There is more work to be done, however, such as integration into GUI builders, which almost exclusively serialize to Java code right now. It’s not enough to store a GUI in XML; there is still logic

“ J2SE should move its focus away from the glass and down into the application framework space ”

It was a sobering discussion for me, because it made me realize one difference between server people and client people. In the client space we’re often obsessed with the GUI toolkit and widget APIs. The server heads, however, seem more focused on the application itself – the transactional nature of the program, the workflow of tasks and data, and issues with multiuser persistence and scalability. Architectures such as Struts, EJB, or JavaServer Faces tackle these issues by providing frameworks that power the application, whereas analogous client initiatives seem lacking. GUI builders still provide palettes of basic JFC components, but when it comes to the question of building a scalable real-world application, the user is often left to reinvent the wheel rather than have these frameworks provided out of the box with J2SE.

Several good ideas came out of the ensuing conversation with the customer regarding things they wanted to see formalized.

## Separate UI and Back-End Logic

It makes sense from an object-oriented viewpoint to separate the UI from the back-end logic, but why not have this architected into the framework. The GUI could be separately serialized into some kind of XML format to allow easy transmission across HTTP, and arguably easier construction for people who preferred markup languages to writing Java Swing code. Other issues such as providing different GUIs for

that is going to occur on the client that needs to be written, so there should be a way that the GUI can be deserialized into the Java client piece and elements of it can be easily accessed – sort of like a JAXB for named Swing components or maybe some kind of XPath solution. While proxy classes can be created for communicating with the Web service, there are quite a lot of code steps to go from WSDL to a usable Java API for the GUI. Some of this could be abstracted out into reusable components that provided dynamic bindings and configurable behavior. Right now the JFC includes graphical component classes, but why not extend it to have more JavaBeans that are less concerned with presentation, but instead know how to broker a conversation between the UI and the data and/or transport layer. One of the projects that I hope might be able to provide these kind of pluggable components is Java Desktop Network Components (JDNC) that offer the promise of an easily customizable API for people to create the middleware components currently missing in Swing. Whatever the answer, I think that J2SE should move its focus away from the glass and down into the application framework space. The solution should not be demoware, however, and should become the foundation framework for users wanting to build large, scalable client/server applications. So instead of building wheels, developers can focus more on their domain-specific logic and less on plumbing and glueware. ☺

Joe Winchester is a software developer working on WebSphere development tools for IBM in Hursley, UK.

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



# WebServices

.NET J2EE XML JOURNAL

LEARN WEB SERVICES. GET A NEW JOB !

## SUBSCRIBE TODAY TO THE WORLD'S LEADING WEB SERVICES RESOURCE

The Best  
**.NET**  
Coverage  
Guaranteed!

Get Up to Speed with the Fourth Wave in Software Development

- **Real-World Web Services:** XML's Killer App!
- How to Use **SOAP** in the Enterprise
- Demystifying **ebXML** for success
- **Authentication, Authorization, and Auditing**
- **BPM** - Business Process Management
- Latest Information on **Evolving Standards**
- Vital technology **insights** from the nation's leading Technologists
- Industry **Case Studies** and **Success Stories**
- Making the Most of **.NET**
- **Web Services Security**
- How to Develop and Market Your Web Services
- **EAI** and Application Integration Tips
- **The Marketplace:** Tools, Engines, and Servers
- Integrating **XML** in a Web Services Environment
- **Wireless:** Enable Your **WAP** Projects and Build **Wireless Applications** with Web Services!
- Real-World **UDDI**
- **Swing-Compliant** Web Services
- *and much, much more!*

Only \$69.99 for  
1 year (12 issues)\*

\* Newsstand price \$83.88 for 1 year

Subscribe online at  
www.wsj2.com or  
call 888 303-5252

\*Offer subject to change without notice



# Turning Components into Domain GUI Objects (DGO)

by Ted M. Young

## Improving your design

**W**hen creating user interfaces for a data entry application (as opposed to one where the user is directly manipulating graphics, such as a network diagram designer), a typical scenario is to create some containers, instantiate their layout managers, and add some components such as JLabels, JTextFields, or JButtons. When the application needs to gather the information entered by the user, it accesses the components directly. Figure 1 shows a screen with a table of information (a book inventory) and a search area that allows users to find one of the books by various criteria specified in the fields.

To find a book with a given word in its title or description, the user enters the query criteria and presses the “Search” button. After the query has executed, the matching items in the table are then highlighted. Listing 1 shows a typical example of how the code can be written where BookSearchPanel.java creates the GUI components, then adds an action listener to the button that, when pressed, calls a doSearch() method that uses the current component’s values to create a search string. (Listings 1–4 can be downloaded from [www.sys-con.com/java/sourcecc.cfm](http://www.sys-con.com/java/sourcecc.cfm).) This article shows the limitations of this coding technique and how to improve it by using Domain GUI Objects.

### Issues with a Typical Solution

In Listing 1 there is very tight coupling between the Swing components and how the search properties are determined. To find out if the Title should be searched, the search code directly invokes isSelected() on the checkbox.

```
boolean isSearchInTitle = m_searchInTitleCheckBox.isSelected();
```

A problem could arise if the user interface changed – the columns to be searched would be specified by selecting items in a JList or highlighting columns in the JTable. The search code would then have to be modified to use the different

components, when all that changed is how the GUI is being presented to the user. The explicit knowledge the search logic needs of the presentation components can be improved upon by decoupling the two.

Another reason why Listing 1 is so poor is that it’s not the responsibility of the search code to determine what the query options are. That logic should be elsewhere, and only the information on what the options are (not how they’re represented) should be required by the search code. By separating the two, it’s possible to add features such as allowing the user to save searches to replay later. Also, what about testing the search code (because I know all of you unit test your code, right?). The “quick and dirty,” tightly coupled implementation is looking less attractive now.

### Loosening the Coupling

There are a couple of choices for improving the design, but I’ll focus on making the “bunch o’ components” into a black box that represents a piece of the application domain. I call this a Domain GUI Object or DGO. The goal is to turn these components into a self-contained object that has a queryable interface for its state. That way, instead of invoking isSelected() on the “Title” checkbox directly, the code calls isSearchInTitle() on the DGO. Listing 2 shows how the BookSearchDgo extends JPanel and has a constructor that accepts an ActionListener that is called back when the search is activated. The query details are retrieved through get methods.

```
public boolean isSearchInTitle() {
    return m_searchInTitleCheckBox.isSelected();
}
```

The presentation logic is encapsulated in BookSearchDgo, which hides its implementation from the code that performs the actual search. This search code can be put in a separate

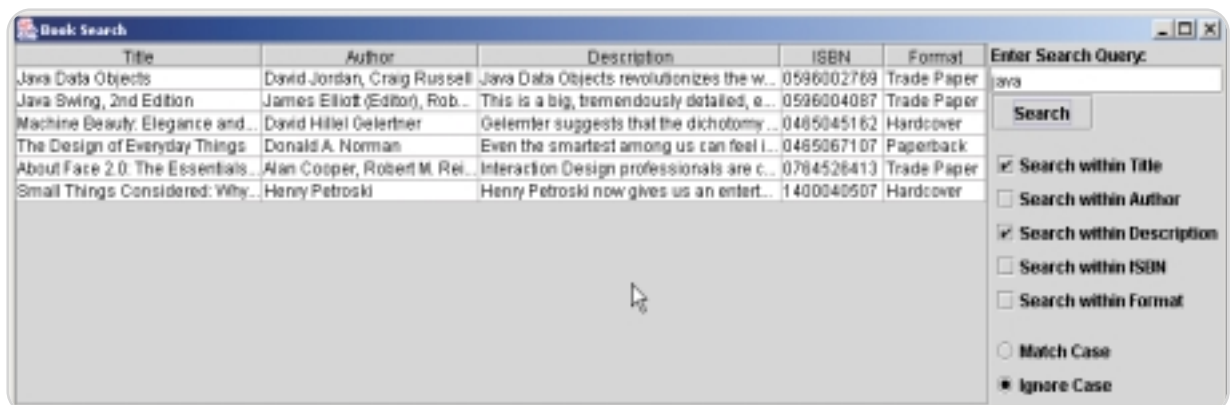


Figure 1 Searchable book inventory with radio buttons for matching case

Ted M. Young is a staff software engineer at eBay, Inc., leading the development of internal desktop tools using Java and Swing technologies. He also cofounded LearningPatterns.com in 1996, the first Java training company in the U.S.

[tmy@jitterpig.com](mailto:tmy@jitterpig.com)

class (BookSearch.java, see Listing 3) that registers itself with the BookSearchDgo (using an ActionListener passed into the DGO's constructor), and when the "Search" button is clicked it queries the BookSearchDgo through its public interface and proceeds on its merry way.

```
public class BookSearch implements ActionListener {
    private BookSearchDgo m_bookSearchDgo;
    public BookSearch() {
        m_bookSearchDgo = new BookSearchDgo( this );
    }
    public void actionPerformed( ActionEvent e ) {
        doSearch();
    }
    private void doSearch() {
        boolean isSearchInTitle = m_bookSearchDgo.isSearchInTitle();
```

Having the BookSearch class instantiate the BookSearchDgo isn't a great way to decouple logic, but the discussion of what creates what and when will have to wait for a future column.

If the way the GUI appears is altered (see Figure 2), such as the Match Case radio buttons becoming a single checkbox, the interface remains the same and the search code is unaltered. This is the advantage of separating the GUI from the search or business logic.

### Loosening the Coupling Even More

Now we feel great: the user interface can change and the search code will remain the same. Uh oh, there's that change of requirements again since the users want the search to be updated in real time. As they type the search text and check the checkboxes, they want the search to be updated immediately without clicking on a "Search" button. Now what?

The solution, shown in Listing 4, is to loosen the coupling at the point where the search code assumed that it was waiting for the "Search" JButton to be clicked (and therefore the ActionListener to be called). Instead of waiting for a button click, the search code performs the search when a more "functional" or "semantic" event occurs. For this an UpdateSearchListener interface is created with a single callback updateSearch() callback method.

```
interface UpdateSearchListener {
    void updateSearch( BookSearchDgo bookSearchDgo );
}
```

The BookSearchDgo accepts an UpdateSearchListener in its constructor

```
public BookSearchDgo(UpdateSearchListener updateSearchListener)
```

and the BookSearch class implements UpdateSearchListener so it's no longer reliant on an ActionPerformed event:

```
public class BookSearch implements UpdateSearchListener {
```

Now the search code doesn't know, or for that matter care, how or why the updateSearch() method was invoked. It could have been that the "Search" button was clicked, one of the checkboxes checked, or just the passage of a specific interval of time. All that the search logic knows is that it needs to update the query, which it still does by getting the search parameters from the DGO that are conveniently passed directly via the updateSearch() method.



Figure 2 Modified user interface to use check box instead of radio buttons for matching case

### Lessons Learned

Tight coupling is bad; loose coupling (up to a point) is good. Tightly coupled code is hard to test, hard to reuse, and easier to break. We've seen two ways to loosen coupling:

1. **Raise the level of abstraction:** Instead of the search code accessing the state of checkboxes, it invokes methods that return state; instead of using ActionEvents, we use UpdateSearchEvents.
2. **Reallocate responsibilities:** Let the DGO collect the information from the GUI and then pass the information to the search code in a compact "Transfer Object" that has only the search query data.

### The Next Steps

In future columns, I'll show how to handle continuing changes in the requirements that affect the GUI: the items that match the search query need to be highlighted; only items that match the search query are shown (filtering); how to switch between two search modes – easy and advanced; and the need to support searching over a large number of items without affecting performance (or at least perceived performance). ☺



# Using Color Technology in Java

by John Chamberlain

## Heighten the visual impact of your application

**B**righten up your dreary application by learning how to use Java to create and manage color. Using a color picker makes it easy to define exact colors with alpha transparency.

For a developer there are two situations in which color comes up: when you are creating the colors for your interface and when a user can define colors or manipulate images as part of your application's functionality. To work with color effectively, you should know the basics of color management and color definition.

### Color Management

Color technology comes in three main types: input devices (like scanners), display devices, and output devices (printers). The possible color range, called the "gamut," differs from device to device. Because of this the colors you see in an on-screen image will be different from those of a scanned or printed image. Controlling the color translation between different environments is what color management is all about.

You can do this translation using the Java 2D API's `ColorSpace` and `ICC_Profile` classes. First load the profiles for the two different environments. For the display you can use the built-in RGB profile that is the default. For a device like a scanner or printer, you'll usually need to load a color profile from a file from the manufacturer. For each of the profiles generate a `ColorSpace` object. `ColorSpace` has two methods: `toCIEXYZ` and `fromCIEXYZ`. The CIEXYZ color space is sort of a master color model that serves as the medium for translation. You can translate any color or image from one color space to another by translating it first to CIEXYZ, then to the target color space.

If you are programming for display only, that's about all you need to know about color management, but if you have color input/output in your application, you may want to learn more (see the resources section at the end of the article).

### Picking Color

When users define a color, they need to have a dialog called a color picker. Unfortunately, the color picker that comes in the Swing package is somewhat inadequate. For one thing it has no support for alpha transparency. Also, it's hard to compare colors and see exact hues. Figure 1 shows a color picker that you can download from JDJ's Web site ([www.sys-con.com/java/sourcecfm](http://www.sys-con.com/java/sourcecfm)). This color picker allows your users (and you) to zero in on colors more easily and includes alpha transparency. Notice that the picker has two boxes: selected color versus compare color, which allow the user to compare two colors against each other. When the selection area is clicked, the compare and selection colors swap places. This is an efficient system for exploring and creating color.

An important consideration for color picking is that you should stick to the HSB (Hue, Saturation, Brightness) system rather than the monitor-oriented RGB encoding. It's much easier to define colors by starting with a hue and then modifying its brightness (shading) and saturation (tinting) than by trying to do an RGB mix. Table 1 shows the basic HSB combinations.

### GUI Color Schemes

One way to increase the vitality of your application's interface is to create a logical color scheme. The problem with the typical battleship gray is low contrast, so in many cases there will be elements that are difficult for the user to read. Also it simply does not look that good; it's boring and monotonous. Many developers are leery of using colors because of the difficulty in finding a good matching set. You can overcome this problem by using the same combination books graphic designers use. I've listed such a swatch book in the resources section.

Even without using a swatch book there are some basic rules you can apply when creating a scheme. In general, it's best to use a two- or three-color scheme plus highlights. In a two-color scheme, it's a good idea to pick two complementary colors. In a three-color scheme start off with a triad (three hues spaced equally). To create highlights decrease the saturation of your base colors.

Other things to be aware of are the distance effect and contrast. When you look from afar, darker colors such as blue fade to black faster; red and blue are inherently darker than yellow and green. Maximizing contrast is a good policy but having black on white can be harsh. Using a pale yellow background may provide better readability. If you need large text to be read from a distance, for example, in a kiosk display, add a red one-pixel outline to the letters to increase contrast. ☺

### Resources

- Fraser, B., Murphy, C., and Bunting, F. (2003). *Real World Color Management*. Peachpit Press.
- Chijiwa, H. (1987). *Color Harmony*. Rockport Publishers.

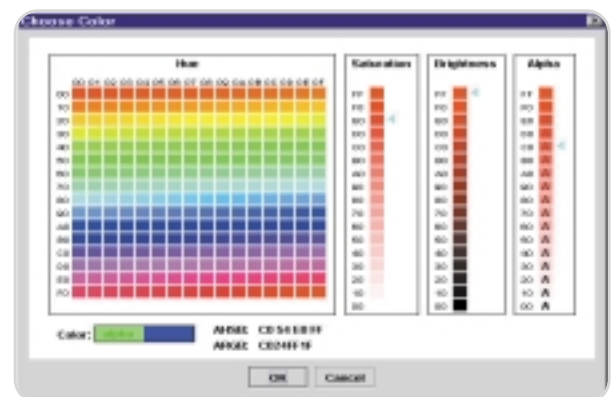


Figure 1 This picker provides good color definition

| Color Type | Saturation | Brightness |
|------------|------------|------------|
| Black      | Any        | 0%         |
| White      | 0%         | 100%       |
| Gray       | 0%         | 1-99%      |
| Hue        | 100%       | 100%       |
| Shade      | 100%       | 1-99%      |
| Tint       | 1-99%      | 100%       |
| Tone       | 1-99%      | 1-99%      |

Table 1 Basic HSB combinations



John Chamberlain,

a developer in the Boston area, works at OPeNDAP.org. He holds an MS in computer science, is a frequent contributor to technical journals, and has spoken at JavaOne. Stop by his Web site at <http://johnchamberlain.com>.

[jdj@johnchamberlain.com](mailto:jdj@johnchamberlain.com)

# SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

## AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!\*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS



### 3-Pack

Pick any 3 of our magazines and save up to **\$275<sup>00</sup>**  
 Pay only \$175 for a 1 year subscription plus a FREE CD

- 2 Year – \$299.00
- Canada/Mexico – \$245.00
- International – \$315.00

### 6-Pack

Pick any 6 of our magazines and save up to **\$350<sup>00</sup>**  
 Pay only \$395 for a 1 year subscription plus 2 FREE CDs

- 2 Year – \$669.00
- Canada/Mexico – \$555.00
- International – \$710.00

### 9-Pack

Pick 9 of our magazines and save up to **\$400<sup>00</sup>**  
 Pay only \$495 for a 1 year subscription plus 3 FREE CDs

- 2 Year – \$839.00
- Canada/Mexico – \$695.00
- International – \$890.00

**CALL TODAY! 888-303-5282**

**Pick a 3-Pack, a 6-Pack or a 9-Pack**

3-Pack     1YR     2YR     U.S.     Can/Mex     Intl.

6-Pack     1YR     2YR     U.S.     Can/Mex     Intl.

9-Pack     1YR     2YR     U.S.     Can/Mex     Intl.

**TO ORDER**

- Choose the Multi-Pack you want to order by checking next to it below.
- Check the number of years you want to order.
- Indicate your location by checking either U.S., Canada/Mexico or International.
- Then choose which magazines you want to include with your Multi-Pack order.

**MX Developer's Journal**

|                                    |                    |                             |
|------------------------------------|--------------------|-----------------------------|
| U.S. - Two Years (24) Cover: \$143 | You Pay: \$49.99 / | Save: \$167 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$72   | You Pay: \$29.99 / | Save: \$60                  |
| Can/Mex - Two Years (24) \$168     | You Pay: \$79.99 / | Save: \$137 + FREE \$198 CD |
| Can/Mex - One Year (12) \$84       | You Pay: \$49.99 / | Save: \$40                  |
| Intl' - Two Years (24) \$216       | You Pay: \$89.99 / | Save: \$127 + FREE \$198 CD |
| Intl' - One Year (12) \$108        | You Pay: \$59.99 / | Save: \$30                  |
| Digital Edition - One Year (12)    | You Pay: \$19.99   |                             |

**Linux World Magazine**

|                                    |                     |                            |
|------------------------------------|---------------------|----------------------------|
| U.S. - Two Years (24) Cover: \$143 | You Pay: \$79.99 /  | Save: \$63 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$72   | You Pay: \$39.99 /  | Save: \$32                 |
| Can/Mex - Two Years (24) \$168     | You Pay: \$119.99 / | Save: \$48 + FREE \$198 CD |
| Can/Mex - One Year (12) \$84       | You Pay: \$79.99 /  | Save: \$4                  |
| Intl' - Two Years (24) \$216       | You Pay: \$176 /    | Save: \$40 + FREE \$198 CD |
| Intl' - One Year (12) \$108        | You Pay: \$99.99 /  | Save: \$8                  |

**Java Developer's Journal**

|                                    |                     |                            |
|------------------------------------|---------------------|----------------------------|
| U.S. - Two Years (24) Cover: \$144 | You Pay: \$89 /     | Save: \$55 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$72   | You Pay: \$49.99 /  | Save: \$22                 |
| Can/Mex - Two Years (24) \$168     | You Pay: \$119.99 / | Save: \$48 + FREE \$198 CD |
| Can/Mex - One Year (12) \$84       | You Pay: \$79.99 /  | Save: \$4                  |
| Intl' - Two Years (24) \$216       | You Pay: \$176 /    | Save: \$40 + FREE \$198 CD |
| Intl' - One Year (12) \$108        | You Pay: \$99.99 /  | Save: \$8                  |

**Web Services Journal**

|                                    |                    |                            |
|------------------------------------|--------------------|----------------------------|
| U.S. - Two Years (24) Cover: \$168 | You Pay: \$99.99 / | Save: \$68 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$84   | You Pay: \$69.99 / | Save: \$14                 |
| Can/Mex - Two Years (24) \$192     | You Pay: \$129 /   | Save: \$63 + FREE \$198 CD |
| Can/Mex - One Year (12) \$96       | You Pay: \$89.99 / | Save: \$6                  |
| Intl' - Two Years (24) \$216       | You Pay: \$170 /   | Save: \$46 + FREE \$198 CD |
| Intl' - One Year (12) \$108        | You Pay: \$99.99 / | Save: \$8                  |

**.NET Developer's Journal**

|                                    |                    |                            |
|------------------------------------|--------------------|----------------------------|
| U.S. - Two Years (24) Cover: \$168 | You Pay: \$99.99 / | Save: \$68 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$84   | You Pay: \$69.99 / | Save: \$14                 |
| Can/Mex - Two Years (24) \$192     | You Pay: \$129 /   | Save: \$63 + FREE \$198 CD |
| Can/Mex - One Year (12) \$96       | You Pay: \$89.99 / | Save: \$6                  |
| Intl' - Two Years (24) \$216       | You Pay: \$170 /   | Save: \$46 + FREE \$198 CD |
| Intl' - One Year (12) \$108        | You Pay: \$99.99 / | Save: \$8                  |

**XML-Journal**

|                                    |                    |                            |
|------------------------------------|--------------------|----------------------------|
| U.S. - Two Years (24) Cover: \$168 | You Pay: \$99.99 / | Save: \$68 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$84   | You Pay: \$69.99 / | Save: \$14                 |
| Can/Mex - Two Years (24) \$192     | You Pay: \$129 /   | Save: \$63 + FREE \$198 CD |
| Can/Mex - One Year (12) \$96       | You Pay: \$89.99 / | Save: \$6                  |
| Intl' - Two Years (24) \$216       | You Pay: \$170 /   | Save: \$46 + FREE \$198 CD |
| Intl' - One Year (12) \$108        | You Pay: \$99.99 / | Save: \$8                  |

**WebLogic Developer's Journal**

|                                    |                     |                             |
|------------------------------------|---------------------|-----------------------------|
| U.S. - Two Years (24) Cover: \$360 | You Pay: \$169.99 / | Save: \$190 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$180  | You Pay: \$149 /    | Save: \$31                  |
| Can/Mex - Two Years (24) \$360     | You Pay: \$179.99 / | Save: \$180 + FREE \$198 CD |
| Can/Mex - One Year (12) \$180      | You Pay: \$169 /    | Save: \$11                  |
| Intl' - Two Years (24) \$360       | You Pay: \$189.99 / | Save: \$170 + FREE \$198 CD |
| Intl' - One Year (12) \$180        | You Pay: \$179 /    | Save: \$1                   |

**ColdFusion Developer's Journal**

|                                    |                     |                            |
|------------------------------------|---------------------|----------------------------|
| U.S. - Two Years (24) Cover: \$216 | You Pay: \$129 /    | Save: \$87 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$108  | You Pay: \$89.99 /  | Save: \$18                 |
| Can/Mex - Two Years (24) \$240     | You Pay: \$159.99 / | Save: \$80 + FREE \$198 CD |
| Can/Mex - One Year (12) \$120      | You Pay: \$99.99 /  | Save: \$20                 |
| Intl' - Two Years (24) \$264       | You Pay: \$189 /    | Save: \$75 + FREE \$198 CD |
| Intl' - One Year (12) \$132        | You Pay: \$129.99 / | Save: \$2                  |

**Wireless Business & Technology**

|                                    |                    |                            |
|------------------------------------|--------------------|----------------------------|
| U.S. - Two Years (24) Cover: \$144 | You Pay: \$89 /    | Save: \$55 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$72   | You Pay: \$49.99 / | Save: \$22                 |
| Can/Mex - Two Years (24) \$192     | You Pay: \$139 /   | Save: \$53 + FREE \$198 CD |
| Can/Mex - One Year (12) \$96       | You Pay: \$79.99 / | Save: \$16                 |
| Intl' - Two Years (24) \$216       | You Pay: \$170 /   | Save: \$46 + FREE \$198 CD |
| Intl' - One Year (12) \$108        | You Pay: \$99.99 / | Save: \$8                  |

**WebSphere Developer's Journal**

|                                    |                     |                             |
|------------------------------------|---------------------|-----------------------------|
| U.S. - Two Years (24) Cover: \$360 | You Pay: \$169.99 / | Save: \$190 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$180  | You Pay: \$149 /    | Save: \$31                  |
| Can/Mex - Two Years (24) \$360     | You Pay: \$179.99 / | Save: \$180 + FREE \$198 CD |
| Can/Mex - One Year (12) \$180      | You Pay: \$169 /    | Save: \$11                  |
| Intl' - Two Years (24) \$360       | You Pay: \$189.99 / | Save: \$170 + FREE \$198 CD |
| Intl' - One Year (12) \$180        | You Pay: \$179 /    | Save: \$1                   |

**PowerBuilder Developer's Journal**

|                                    |                     |                             |
|------------------------------------|---------------------|-----------------------------|
| U.S. - Two Years (24) Cover: \$360 | You Pay: \$169.99 / | Save: \$190 + FREE \$198 CD |
| U.S. - One Year (12) Cover: \$180  | You Pay: \$149 /    | Save: \$31                  |
| Can/Mex - Two Years (24) \$360     | You Pay: \$179.99 / | Save: \$180 + FREE \$198 CD |
| Can/Mex - One Year (12) \$180      | You Pay: \$169 /    | Save: \$11                  |
| Intl' - Two Years (24) \$360       | You Pay: \$189.99 / | Save: \$170 + FREE \$198 CD |
| Intl' - One Year (12) \$180        | You Pay: \$179 /    | Save: \$1                   |

\*WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today [www.sys-con.com/2001/sub.cfm](http://www.sys-con.com/2001/sub.cfm)





Ted Neward

# No Fluff, Just Stuff: A Just In Time Conference

An interview with Jay Zimmerman and Ted Neward

Interviewed by  
Kirk Pepperdine

**L**ast year I was introduced to the No Fluff, Just Stuff (NFJS) Software symposium ([www.nofluffjuststuff.com](http://www.nofluffjuststuff.com)) in Atlanta. It was a high-quality experience in every aspect. The speakers delivered more than technical information, they delivered an insight into what was happening in our industry. The Dallas edition of NFJS (Lone Star Software Symposium) maintained (if not exceeded) this high standard and it too delivered on the promise of a small and very personal experience where participants and presenters mix throughout the three-day event.

NFJS was organized by Jay Zimmerman in 2001 after a number of conversations he had with members of the Boulder/Denver Java Users Groups. I was able to catch up with Jay between sessions. During our conversation, you could see that his meticulous attention to detail was driven by his passion to create a forum that has garnered a lot of respect in the industry. You can see that passion in his words.

**JDJ:** Jay, what motivated you to start NFJS?

**Jay:** I am heavily involved in the Boulder Java User Group and was talking to a number of people there. We thought it would be great if we could have a technically focused event right here in Denver. We looked at the national conference model and said: what if we could localize it, focus on technical content, and find a way to attract teams of developers. We wanted to focus on development teams because we've found that the trickle-down effect that everyone was looking for from national conferences wasn't really happening. National conferences elevated the individual who got to attend. We wanted to elevate the entire team.

We contacted a number of companies and from conversations with them, we decided that a Friday to Sunday model would work the best. That way, companies were willing to allow entire teams to attend, as it didn't hurt their immediate schedules. As it turns out, it also offered a number of side benefits. First, most of the good speakers are independents who are heavily engaged. This schedule gave us access to those people. The other benefit was that we could also attract contractors and consultants as this group has long been ignored. These people take a double hit when they take time out to attend a conference.

There was a little resistance in the beginning, but then things really took off when we demonstrated that we were able to reliably provide a high-level event that is always worthwhile and something to count on. To be a better Java developer is worth one weekend a year of your time.

**JDJ:** How were you able to attract these high-caliber speakers?

**Jay:** We connected the dots with James Duncan Davidson and other top-flight speakers, then things just took off from there. The speakers like the highly interactive nature that is encouraged. Many of the speakers not only speak, they attend the other seminars; they dine with the participants. The whole structure is set up to be very personal and highly interactive. We currently get about three to four speaker requests per month. What we are looking for are speakers with fresh material with a view of where the industry is going.

**JDJ:** Why the lack of vendor presence?

**Jay:** We do have a limited number of sponsorship opportunities for passive marketing. We will distribute software and pamphlets, but what we are striving for is an environment that is friendly, not disruptive. I am proud that we get comments like "no one was selling me something." This get kudos from the developers and it does create a lot of goodwill with vendors.

**JDJ:** Is there a message that you'd like to send?

**Jay:** Yes. What we are trying to offer is something of extraordinary value that allows companies to reward their employees equally. Too often it is only the technical architect who gets to go to a national conference. We, as developers, get closed in our four walls. Once a year it's good to peek out and get a fresh view. And now we're making this easier by expanding to include 24 different stops in the U.S. and Canada.

• • •

*The first lecture that I attended was a spirited talk on decoupling patterns presented by Dave Thomas (The Pragmatic Programmer). Dave's apparent meanderings were actually a very highly structured, methodical, and entertaining presentation on couplings and how to avoid them. During the presentation, Dave proclaimed, "I hate design patterns." His defense to such heresy was that "patterns should not lead your designs." He went on to say that patterns should be used as a standard language that describes elements of your design.*

*In addition to the coverage of development techniques, a number of the seminars focused on Tiger (J2SE 1.5). Glen Vanderburg talked about the concurrent programming features specified in JSR 166 and Ted Neward lectured on custom metadata specified in JSR 175. Ted is an independent software consultant and author who "just loves to talk a lot." I had the opportunity to listen to Ted between sessions. Here's what he had to say.*

**JDJ:** How long have you been involved with NFJS?

**Ted:** I'm still in my first year doing these; IIRC, my first show was in Denver back in May. That said, almost from the very first show, I've felt like I've found a group to which I really



Now  
More  
Than  
Ever  
Before...

JAVA DEVELOPER'S JOURNAL

**JDJ**

Means

**Business**



Subscribe Now!

**\$69.99**  
1 YEAR/  
12 ISSUES

[www.SYS-CON.com/JDJ](http://www.SYS-CON.com/JDJ)

or call

**1-888-303-5282**

**The World's Leading i-Technology Magazine Just Got Better!**

Now with expanded business coverage, *JDJ* is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of IT professionals.



The Most Popular  
**i-Technology Magazine**  
in the World !

**SYS-CON  
MEDIA**

The World's Leading i-Technology Publisher

“belong” – my very first speaker panel with Jason, James, Bruce, Dave, and Stu, among others, was an absolute *blast* and definitely one of my favorite memories as a speaker. The speaker panel at these shows is definitely not something to be missed. It’s two parts stand-up comedy, two parts pithy wisdom, one part absolute enlightenment, and that’s for someone sitting *on* the panel – I can’t imagine what it must be like to sit out there in the audience. I’m absolutely honored by the company I’m with up there; I just provide the comic relief.

*JDJ: Is there a message that you wish to impart or leave the audience with?*

**Ted:** Honestly – a sense of “questioning”; so many times developers simply take the marketing material spoon-fed to them and accept it as doctrine and gospel without ever really stopping to think about what’s going on under the hood. (Frankly, who has time to do anything but take them at face value?) But this is a correctable state of affairs simply by forcing Java developers to slow down for a moment and *think* the situation through. Ever stop to think how an RMI call actually works? Going through the mental exercise tends to yield some interesting insights that, lo and behold, yield some great suggestions about how to make a distributed object system *not* suck. If I can get the attendees to start thinking about these things, particularly with respect to upcoming technology

seems the cat has already been let out of the bag. That said, though, I have to prefix any talk I give on the metadata attributes stuff with a standard disclaimer (and the attendees are required to swear an oath to this effect) that if you build a production system based on what’s presented here, you’re liable to be pounded repeatedly with a whiffle bat until you agree it’s a bad idea to do so.

By the time you read this, we should have released a Public Draft, but until that actually makes the streets, everything’s up for grabs. There’s also, I believe, a large misperception among the Java community at large about what metadata is and is for, mostly driven by systems claiming to be “aspect-oriented” that use attributes to convey the sort of interception that should take place against those methods; the perception is that JSR 175 is going to somehow codify that and make it a standard part of the language. Nothing could be further from the truth, and given the venue that I’m able to command through the NFJS shows, I’ve got a responsibility, I believe, to make sure the *right* message goes out to the world.

*JDJ: Just what is metadata and how do you see it being used?*

**Ted:** Officially, it’s “data about data.” In an RDBMS, it’s the tables that describe the tables in your database instance. In Java, metadata is the Reflection model, the classes, fields, and methods that you build. Among other uses, it’s what allows

“Many times developers simply take the marketing material spoon-fed to them and accept it as doctrine”

gies (When are Web services just like EJBs? When you use them like EJBs. When are they better than EJBs? When you *stop* using them like EJBs), then I’ve done a good day’s work. Teach a man to fish...

*JDJ: What do you draw from this symposium?*

**Ted:** More than I’m able to put into it, that’s for sure. I get to rub shoulders (both during the show and at the traditional Saturday night dinner-and-a-movie gathering we do at every show) with guys like Dave Thomas, Jason Hunter, James Duncan Davidson, Erik Hatcher, and, of course, some of my DevelopMentor buddies like Stu Halloway. I learn something new every time. On top of which I get to meet folks I’ve never met in person before: guys like Mike Clark, Bob Lee, Glenn Vandenburg, all of whom teach me even more, and even meet in person the folks I’ve met before but only over e-mail, like Bruce Tate, who’s got a *great* story about how we came to know each other through Manning. :-)) More than that, though, the small-scale nature of the show gives me the opportunity to poll the audience and actually give them a chance to offer insights on the material I’m presenting – it seems like almost every show I have somebody telling me some interesting little tidbit about software that I didn’t know before. It forces me to rethink my positions on software, and that’s probably the best benefit I can possibly imagine from a show.

*JDJ: What prompted you to talk about the metadata features in Tiger?*

**Ted:** Officially, the stuff I’m discussing in the JSR 175 talk is still under wraps, but considering that our Expert Group lead, Josh Bloch, and a few others have been engaging in Webcasts and other discussions to promote the new features of Tiger, it

technologies like Java’s Object Serialization and Hibernate to be able to reach into a Java object, discover its contents, and scoop out (or shove in) the values stored in those fields. It’s powerful stuff.

*JDJ: You mention in your talk that this will fundamentally change the Java landscape. Can you describe some of the changes that you expect to see?*

**Ted:** A lot of mechanisms currently exist to try and provide extensions to Java’s standard metadata through various means: J2EE uses deployment descriptors to offer custom metadata about transactional affinity (EJB) and servlet mappings (servlets), while JDO uses an XML “persistence descriptor” to describe how a JDO enhancer should modify your compiled classes to provide transparent persistence capabilities. Both of these things, along with many others, can and should be codified into your compiled binary, such that you can obtain that data without having to resort to mechanisms stored outside of the source code. Storing it externally to your source forces another point of maintenance and introduces the possibility that a mismatch can occur; this chance is greatly reduced, although not entirely eliminated, when the “descriptive” data is stored inside the source file it references. Although it’s out of scope for our JSR to define some annotations outside of those needed for the core functionality, I can easily imagine several annotations that would have an immediate and powerful impact on the community:

(\*) JavaBeans “Bean”, “Property”, “Method”, and “Event” annotations, rather than naming patterns

(\*) XML Serialization enhancements (a la .NET's XmlSerializer attributes)

(\*) Outright replacement of such "marker interfaces" as Remote and Serializable

(\*) Declarative security constraints enforced by the JVM, rather than explicit Permission checks at the start of each method

and so on. More and more "stuff" will be described using annotations rather than external "descriptor" files, and having this functionality built into the language and compiler will release tools like XDoclet from trying to act like metadata attribute systems and go back to their core focus, which is code generation.

Again, so long as your readers understand that reading this code example implies they're taking the whiffle-bat oath, using an annotation would look something like this:

```
import com.javageeks.fictitious_persistent_library.*;

@Serializable @PersistentClass(tableName="PERSON_TBL")
public class Person
{
    @PersistentField(columnName="first_name")
    private String firstName;

    @PersistentField(columnName="last_name", type="VARCHAR(80)")
    private String lastName;

    // . . .
}
```

*JDJ: What is the "Arms Length API" and do you feel that it is important?*

**Ted:** The "At-Arms-Length API" is our current working term for some kind of API that would allow a tool (pre-processor or postprocessor) to examine metadata without having to load the actual class or its annotations into the JVM. This actually has some powerful ramifications – it means that you'd be able to load into a postcompilation utility (like a JDO enhancer) a class whose annotation types you don't have on your CLASSPATH; the missing annotations would be described, but would not be able to be loaded until you put those annotation type binaries into a ClassLoader that can find them. Unfortunately, it looks like the Expert Group will have to defer defining this particular API for the Tiger release; we're just not sure we can get it done in time to make the JDK 1.5 release date. This is still a "hot" issue for the group though, so I strongly encourage feedback through the e-mail address listed on the 175 page at the JCP site, if readers are strongly in favor (or opposed) to that API. As a matter of fact, I strongly encourage everybody to have a look and see if it makes sense to you – the more data points we as an expert group have, the better we can make decisions that everybody will have to live with.

### Conclusion

*The only disappointment I felt was when I looked at the schedule and realized that I would not be able to attend as many of the lectures as I would have liked. This is a clear testament that this symposium is time well spent. ☺*

• • •

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



**The Leading Magazine for Enterprise and IT Management**

# LinuxWorld Magazine

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward. Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude. Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom. Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *LinuxWorld Magazine* is aimed squarely at providing this group with the knowledge and background necessary to make decisions to utilize the Linux operating system. Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *LinuxWorld Magazine* does not feature low-level code snippets but focuses instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month presents a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

**Regular features include:**

- Advice on Linux Infrastructure
- Detailed Software Reviews
- Migration Advice
- Hardware Advice
- CEO Guest Editorials
- Recruiting/Certification Advice
- Latest News That Matters
- Case Studies

**SAVE 30% OFF!**  
REGULAR ANNUAL COVER PRICE \$71.76  
**YOU PAY ONLY \$49<sup>99</sup>**  
12 ISSUES/YR

**SUBSCRIBE TODAY!**  
[WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)  
OR CALL 1-888-303-5282

**FOR ADVERTISING INFORMATION:**  
CALL 201 802.3020 OR VISIT [WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)

\*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

IDG  
LINUXWORLD® IS THE REGISTERED TRADEMARK OF INTERNATIONAL DATA GROUP, INC.

SYS-CON MEDIA  
LINUXWORLD® IS THE REGISTERED TRADEMARK OF INTERNATIONAL DATA GROUP, INC.

The World's Leading Technology Publisher





# Droplets by Droplets Inc.

Reviewed by  
Somnath Banerjee

**T**ired of using that Web access e-mail client? Going crazy filling out those expense reports using a clunky Web-based system? Missing the superior and responsive desktop application? You're not alone! According to the usability research firm Nielsen Norman Group, "Billions of dollars are wasted yearly in lost productivity as people wait for Web pages to perform duties that could be better handled by a 1984 Macintosh-style GUI application."

The promise of Web architecture (no deployment, large scalability, no firewall opening) is hard to refuse. Yet the wide deployment of HTML-based applications has left end users craving for a responsive and productive user interface. There have been several efforts to combine both the Web and the rich UI (DHTML/ActiveX/Java applets). Unfortunately, these add more pain rather than soothe.

The Droplets application platform, with the User Interface Server and the SDK, enables developers to create software that combines the convenient, instant deployment benefits of the Web with the high usability and rich UI of traditional clients. Droplets GUIs can be dragged directly off of Web pages and onto the desktop, acting as double-clickable icons to launch Web-based apps that look and feel like Windows applications. As there is zero application code on the client machine, Droplets is less prone to security breaches.

## Droplets Overview

Droplets has a downloadable SDK from the Droplets Web site developers zone ([www.droplets.com/developer](http://www.droplets.com/developer)). There is an initial evaluation license and plenty of starter materials (tutorials, samples, documentation) to get started. The first time I installed Droplets and tried it out, I could get to a quick sample of mine in less than two hours. Command-line and Notepad-based development is possible. Droplets also has a plug-in for JBuilder and Eclipse. Easy integration with webMethods' Glue makes it possible to hook up with back-end Web services.

The key components of Droplets are as follows.

### Droplets UI Server

The Droplets UI Server hosts application logic, instructs the client about GUI rendering, manages client connections, and provides GUI updates in response to both user-initiated and server-side events. The UI Server can also integrate with application servers and back-end components like XML-Web services or EJB/CORBA.

### Droplets Client

The client is a lightweight engine transparent to the end user that renders all Droplets applications at the client computer. The Droplets client does require a small, one-time download (~1MB). Thereafter, no application downloads are required and all upgrades are automatic on application start-up. The client has two responsibilities: the presentation of the GUI and the reporting of user events to the UI Server. To accomplish this, a small (1-2KB) text file called a "DRP" is used. It holds basic information like the server URL, the server

port, and the name of the Droplet to present. The Droplets client also has the ability to support a "Dripline", a desktop alert that allows the UI server to contact a client whenever the state of a given Droplet has changed. This allows the client to update the Droplet's desktop icon, e.g., alerting the user of server events like a change in a stock price or an inventory level dipping below a threshold.

### Communication Layer Enables Superior Performance

The Droplets client and server communicates via an optimized communications layer running over TCP/IP sockets. The UI Server instructs the client as to the layout of the applications at start-up, and transmits both server-side and user-initiated updates. Droplets transmit only events and individual field updates, whereas an HTML-based application reloads the entire page for each user event. This makes a Droplets application faster and requires less bandwidth than an equivalent Web application.

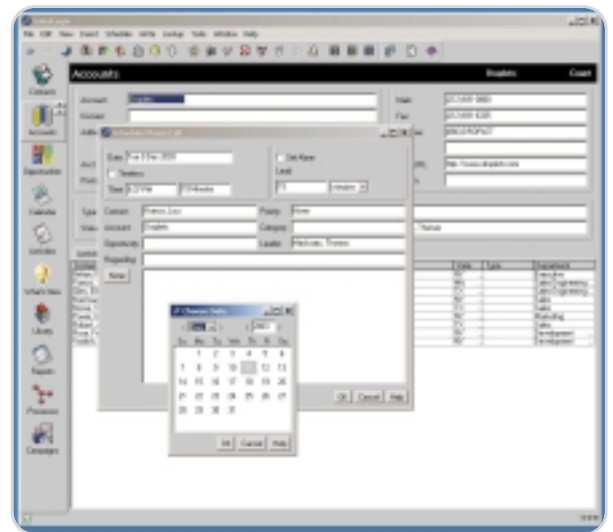


Figure Saleslogix like App Screen

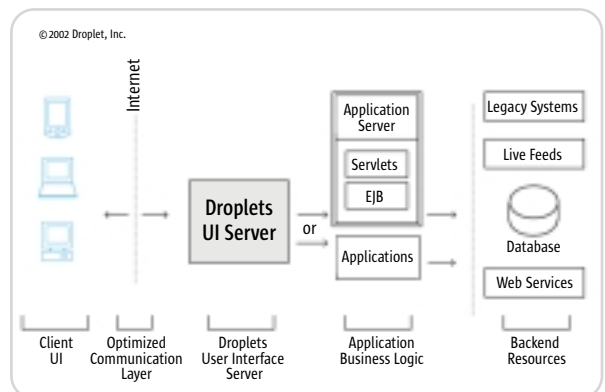


Figure Overall Droplets Architecture

Somnath Banerjee is the founder of MEC Technologies, a provider of Web services and X-Internet-related technology products and services. He has 18 years of experience in the development of software products. Somnath has a BS in EE from the Indian Institute of Technology and an MS in CS. He has authored several technical papers and shares patents.

[sbanerjee@mectechnologies.com](mailto:sbanerjee@mectechnologies.com)

## Droplets SDK

The Droplets Software Development Kit (SDK) is a remote GUI toolkit that allows development using a single standard language. Java and C++ are the current choices. OO-COBOL, C#, and VB should be available in early 2004. The API library is based on Java's AWT package and also incorporates components and interfaces from the Swing package. The Droplets platform provides several infrastructural components such as networking, encryption support, an authentication mechanism, and monitoring. Droplets' server-based architecture allows customization of the look and feel of application instances via "skins," allowing enterprises to blend the Droplet into the look of their other online content.

## A Real-Life Scenario

The following is an example of a real-life scenario: a large chemical company uses a standardized set of key business performance measures. The company's executives use the key measures that span financial, economic, and customer satisfaction data to make critical business decisions. A business intelligence system from COGNOS was used to gain visibility into a number of different business units using various IT systems. There are over 200 business units, and each unit is measured by around 50 measures. This data is currently loaded every month through a complex ETL procedure, then Excel and paper reports are prepared for the management. These reports are available every 30 days and are largely static in nature. However, the existing Web-based GUI was too slow, cumbersome, and user-unfriendly. It became so unwieldy that the management staff preferred paper reports.

MEC Technologies was hired to build a management dashboard to improve the ease of accessing and reporting data for the management staff. While enriching the end-user experience was critical, it was also important to use industry-standard XML Web services to integrate data from BI Systems, back-end DB, and other third-party data sources. Considering the user base, a highly polished and easy-to-use application was desirable.

Using the Droplets Java API, an executive management dashboard was developed in less than two months. Since the entire application was server-based, hooking up with back-end Web services was quite simple. Initially, both Apache Axis and webMethods Glue were used to create Web services proxies and access the measures data. Finally, webMethods Glue was chosen. The user community used NTLM security to get authenticated and then access a URL within the intranet. That URL would launch the Droplets application. Most users remained with the browser-based UI. However, a few advanced users preferred to use the Droplets client. A controlled version was beta tested by the financial analysts community. Once they were satisfied, the application was launched in production for the executive management staff.

## Experience Summary

The overall experience of developing with Droplets and using it was quite encouraging. There were very few platform- and Droplets (SDK)- related roadblocks. Data integration (using Web services) was simple due to the server-side architecture and usage of XML and SOAP-based services. Thanks to the Droplets system and access logs, debugging applications during development and production was easy. Session-specific debugging was also enabled. What was really nice was the post-production extension of the application. An application-level usage reporting module was requested, and it was developed and deployed in less than two weeks. The entire new application extension was launched with minimum downtime.

## Conclusion

The Droplets platform presents a nice alternative for both developers and IT managers. Developers can use their favorite

## Droplets Inc.

636 Avenue of the Americas  
New York, NY 10011  
**Web:** [www.droplets.com](http://www.droplets.com)  
**Phone:** 212 691-0800  
**Fax:** 212 691-6335

## Specifications

- Deploys standalone, or with any J2EE-compliant application server
- Supports Windows NT, Windows 2000, Linux, Solaris
- Programming tools for Java and C++; OO Cobol and VB.Net coming soon
- Integration with Borland JBuilder; Oracle JDeveloper and Eclipse in beta
- Windows, Linux, Unix, wireless devices as standalone apps/within a Web browser
- Integrate with Web services protocols with webMethods' Glue

## Snapshot

### Pros:

- Easy-to-build, highly rich Web-based applications
- Secure application deployment
- Java-based server enables cross-platform support

### Con:

- No offline capability

IDE and use a high-level component-based environment. IT managers can still enjoy their zero-touch deployment.

Microsoft has recognized this space and presents a solid case with smart clients ([www.microsoft.com/net/products/client.asp](http://www.microsoft.com/net/products/client.asp)). Let's compare Droplets with .NET smart clients.

With smart clients, any machine that has a .NET CLR installed can download a .NET assembly and can do anything that the CLR classes and the security framework allow, so it fits the "rich client" aspect quite nicely. On top of this, .NET allows automatic synchronization of code versions. The IEEEXEC component (integrated with IE) keeps a local cache synchronized with the server-side assembly and does delta updates automatically.

Droplets, a Java-based server, is platform independent. Currently it runs on Windows, HP Unix, Sun Solaris, and main-frame Linux. Microsoft .NET remains a Windows family-only solution. This is major for IT shops having disparate systems and trying to put sharper front ends to their traditional back-end systems.

### Smart Clients Promise Offline Capability

The Droplets platform always needs a network connection to the server. However, smart clients are capable of working with local caches (both file system and DB engine based). The promise of being able to work on an application with or without a network connection is huge and will fuel an entire class of on-the-road applications.

However, Droplets has a strong story in the security area. A .NET smart client depends on code being downloaded to the client machine and running on an environment managed by a CLR. It's quite cumbersome to set up the client-side privileges of downloaded code using the Code Access Security (CAS) model. This also leaves much to worry about considering what security door might be left open in downloaded code, exposing it to some serious security breach.

The Droplets platform presents a compelling option for building rich Web applications using a secure and a platform-independent architecture. ☺

## Resource

- *Source code:* [www.sys-con.com/java/sourcec.cfm](http://www.sys-con.com/java/sourcec.cfm) or [www.mectechnologies.com/articles/JDJ/Droplets/Trader.zip](http://www.mectechnologies.com/articles/JDJ/Droplets/Trader.zip)



Onno Kluyt

# From Within the Java Community Process Program

## From proposals to final approvals

**W**elcome to the February edition of the JCP column! Each month you can read about the Java Community Process: newly submitted JSRs, new draft specs, Java APIs that were finalized, and other news from the JCP. This month I'll discuss four new JSRs and a few JSRs that are in the Proposed Final Draft and Public Review, and one final JSR.

### Let's Start with J2ME Technology

A new JSR in this space is JSR 238, Mobile Internationalization API. It proposes to add culturally correct data formatting, and the sorting of text strings and such for MIDlets in a CLDC/MIDP environment, according to its submitter Nokia. The broad support for internationalization that you find in the J2SE platform is not part of the CLDC/MIDP environment because of memory footprint constraints. This optional package will deliver the appropriate level of internationalization functionality to Java-based devices.

JSR 226, also by Nokia, has reached Community Review. This JSR defines an API for scalable 2D vector-based graphics for J2ME technology, building upon the W3C specification for the Scalable Vector Graphics (SVG) format. This JSR was submitted this past summer and has made good progress through the process, reaching this milestone six months later.

what I'd like to discuss now are three new JSRs, one JSR in Public Review, and one final.

BEA and IBM submitted three new JSRs to the Community for approval for development. JSR 235 proposes an API for so-called Service Data Objects (SDO). SDO is roughly based on the Data Transfer Object pattern that many J2EE developers are familiar with. The JSR proposes to standardize Data Objects functionality with regards to changing the history, metadata, XML support, and neutral representation of business data among other things. JSR 236 proposes a timer for the Application Servers specification. It would provide an alternative to the existing `java.util.Timer` package and the `Timer` API in JMX. The third JSR is JSR 237, Work Manager for Application Servers. The submitters propose to provide an API to enable the concurrent execution of application-level work (for example, made up of EJB components or servlets) in J2EE-based environments. This JSR is closely related to JSR 166, which provides such facilities for J2SE 1.5 environments.

JSR 206 works on the next version of the Java API for XML Processing (JAXP), now in its third generation within the JCP, reaching Public Review 10 months after its submittal. There are several new main areas – XML 1.1, name spaces in XML 1.1, Document Object Model level 3, and SAX (Simple API for XML) version 2.0.1.

I'll complete this section on J2EE-related JSRs with the Rule

“ JCP.org's redesigned Web site will make it easier for you to find news about the JCP and about the JSRs ”

### Next, J2SE Technology

Late December 2003, the spec lead and expert group for JSR 127, JavaServer Faces, submitted the Proposed Final Draft for this API. This JSR defines a series of JSP tags and Java classes that enable the developer to build richer and more appealing user interfaces for Web-based applications than is achievable with just JSPs and servlets.

The next JSR I'll discuss just entered Public Review. JSR 200, led by Sun, involves a network transfer format for Java archives. The aim of this JSR is to deliver a dense download format for class files that can create much smaller archives than compressed JSR files. Once final, the technology will be delivered as part of the next version of the J2SE platform, aka “Tiger.”

### Then, J2EE Technology

The J2EE environment has seen the most activity in the last month or so. The J2EE 1.4 JSRs went final, of course, but

Engine API for Java led by BEA. JSR 94 successfully passed the Final Approval Ballot in December. It is BEA's first JSR to become final and thus a “well done, lads!” to BEA. The API aims to reduce the development cost coming from incorporating business logic within applications by capturing this logic in sets of rules upon which operations and calculations can be performed.

### Closing Remarks

By the time you read this column, the JCP.org Web site will look a little different than before. Its redesign will make it easier for you to find news about the JCP and about the JSRs, so you can quickly see if there's something new regarding your favorite JSRs and download the most current drafts. More changes are in the pipeline for the early March time frame, when the launch of JCP 2.6 is planned.

That's it for this month. I'm very interested in your feedback. Please e-mail me your comments, questions, and suggestions. ☺

Onno Kluyt is the director of the JCP Program Management Office, Sun Microsystems.

onno@jcp.org



| Advertiser               | URL                                     | Phone          | Page      |
|--------------------------|---|----------------|-----------|
| BEA Systems              | http://dev2dev.bea.com/workshop1        | 800-817-4BEA   | Cover II  |
| Borland                  | www.go.borland.com/j3                   | 831-431-1000   | 9         |
| Computer Associates      | www.ca.com/lifecycle                    | 631-342-6000   | 13        |
| Compuware                | www.compuware.com                       | 800-COMPUWARE  | 21        |
| Crystal Decisions        | www.businessobjects.com/v10/047         | 800-877-2340   | 19        |
| CTIA Wireless            | www.ctiashow.com                        | 301-694-5243   | 47        |
| Cyanea                   | www.cyanea.com/wsdj/underpar.html       | 877-CYANEA8    | 3         |
| DataDirect               | www.datadirect.com/jdj                  | 800-876-3101   | 7         |
| Dice                     | www.dice.com                            | 877-386-3323   | 31        |
| ESRI                     | www.esri.com/mapobjectsjava             | 909-793-2853   | 35        |
| Extentech                | extentech.com/jdjxls/                   |                | 25        |
| Google                   | www.google.com/cacm                     |                | 39        |
| GreenPoint               | www.webcharts3d.com/demo                | 212-765-6982   | 51        |
| ILOG                     | jviews-info-kit.ilog.com                | 1-800-for-ILOG | 29        |
| Infragistics, Inc.       | www.infragistics.com                    | 800-231-8588   | 14-15     |
| InterSystems             | www.intersystems.com/match3             | 617-621-0600   | 4         |
| iSavix                   | http://isavix.net                       | 703-689-3190   | 45        |
| Java Developer's Journal | www.sys-con.com/java                    | 888-303-5282   | 59        |
| LinuxWorld Magazine      | www.linuxworld.com                      | 800-303-5282   | 61        |
| Mercury Interactive      | www.mercuryinteractive.com/optimizej2ee | 800-837-8911   | 11        |
| NetIQ                    | www.netiq.com/solutions/web             |                | 49        |
| Northwoods Software      | www.nwoods.com/go                       | 800-434-9820   | 43        |
| Oak Grove Systems        | www.oakgrovesystems.com/jdj             | 818-440-1234   | 33        |
| Parasoft Corporation     | www.parasoft.com/jdj_02                 | 888-305-0041   | 23        |
| PortalVU                 | www.basicportal.com/nyc                 |                | 27        |
| Quest Software, Inc.     | http://www.quest.com/jdj                | 800-663-4723   | Cover IV  |
| Software FX              | www.softwarefx.com                      | 800-392-4278   | Cover III |
| SYS-CON Publications     | www.sys-con.com/2001/sub.cfm            | 800-303-5282   | 57        |
| SYS-CON Reprints         | www.sys-con.com                         | 201-802-3026   | 55        |
| WebAppCabaret            | www.webappcabaret.com/jdj.jsp           | 831-464-6941   | 37        |
| Web Services Journal     | www.wsj2.com                            | 888-303-5252   | 53        |
| Wily Technology          | www.wilytech.com                        | 888-GET WILY   | 41        |

**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.

This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

# Next Month

## Industry Perspective

Oracle's SVP Thomas Kurian on the business impact of grid computing



Thomas Kurian

### Clustered Timers

Often, when someone asks how will we scale the Web application we're about to develop, we look at them, smile, and say, "Not a problem - we'll just cluster the application servers." Clustering our application across multiple servers provides us with the ability to handle large volumes of traffic and to scale systems by adding additional servers to the cluster.

### The Perils of Copy-Paste Coding

Copy-paste coding is the practice of copying a portion of existing code, pasting it elsewhere, and modifying this new version to solve a slightly different problem than the original one. This article provides a number of suggestions on how to refactor copy-paste code into cleaner code that's easier to test and maintain.

### Client/Server Is Not the Only Fruit

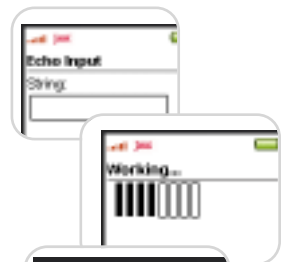
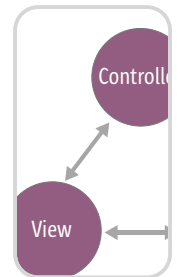
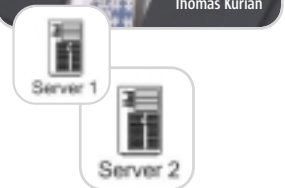
The vast majority of Java enterprise applications are architected along the lines of Sun's original PetStore showcase application. Rather than seeing this as a market stall displaying all the J2EE goods available, developers took it as a blueprint for enterprise applications. This article briefly describes some of the testability shortcomings of the traditional J2EE model and presents some alternatives.

### Extreme Analysis

Whenever I read about XP, everything seems to be known and crystal clear for the customer and the developer. Everybody just has to sit down, write down the user stories or whiteboard sketches of forms or workflows with a few entry fields and some buttons... But what about the business analysis? A set of stories can build a nice set of features to implement and, depending on the analytical, creative, and communication skills of your developers, I believe you can build a nice system if it's implemented story by story.

### Building a Connected MIDlet, Part 2

In Part 2 in this series devoted to developing a connected MIDlet, we'll refine the design of the application, add a few features that are essential to any connected MIDlet, and implement exception handling. By the end of this article, you'll know how to break up a single-class application into better partitioned packages and classes, invoke a new method in a background thread, interrupt an ongoing background thread, and gracefully handle application-specific exceptions.



# Offshore Outsourcing: Magic Bullet or Dirty Word?

It all depends on your perspective



Jack Martin

In the world of IT, outsourcing is either the dirtiest word you can utter or a brilliant one; it's all about who says it to whom and where it is said.

No matter who uses it, it is a word most often said in private. When corporate managers use the word, it is always mentioned in a most confidential fashion as a potential cost-cutting tactic, a magic bullet to increase margins.

When technical people use the word in public it is always with a hushed tone, as if speaking it aloud would give management the idea. In private it is discussed as if it were the greatest evil ever to befall the world, a faceless monster from far away.

programmer and I can wear anything I want to work," which was taken to the extreme by some people. Management was wearing suits and in contrast the programmers looked like they came from some alien planet. The more outrageous the better.

From there, showing up at work at the same time as the rest of the staff became optional – the later the better – with the excuse that they were up all night writing code. It's true that a lot of code writers were up late at night writing code, but often not for their day job. An awful lot of people were busy writing code at night for dotcom business plans with IPO dollars in their dreams, while the more pragmatic moonlighted for

“ Outsourcing can be an extremely complex and complicated undertaking. Each piece of the process needs to be considered with great care and executed with precision. There is little margin for error halfway around the world. Once a company decides to outsource its code, programmers know their days are numbered. It's just a question of when the ax will fall”

Jack Martin, editor-in-chief of *WebSphere Journal*, is cofounder and CEO of Simplex Knowledge Company, publisher of America's Job Market, the only independent source for unbiased advice about careers. Simplex developed the first remote video transmission system designed specifically for childcare centers and the world's first diagnostic-quality ultrasound broadcast system. Jack is coauthor of the upcoming book, *Understanding WebSphere*, from Prentice Hall.

jack@sys-con.com

The reality falls somewhere in the middle.

Outsourcing can be an extremely complex and complicated undertaking. Each piece of the process needs to be considered with great care and executed with precision. There is little margin for error halfway around the world. Once a company decides to outsource its code, programmers know their days are numbered. It's just a question of when the ax will fall. It is also just a matter of time before a major project goes completely out of control and craters, leaving hapless managers thrashing about with a project team in India.

So today we have corporate managers blindly sending work halfway around the world – and an endless drain of jobs overseas. Who came up with this latest corporate fad? How we got here is an interesting paradox.

Let's take a walk down memory lane. During the dot-com days, American code writers as a group became major prima donnas. It all started with the attitude, "I'm a pro-

other companies desperate for anyone who could write code.

Then the "I have to bring my dog to work" concept started. All of a sudden a menagerie of pets started showing up at work. Further, some programmers demanded and received trampolines. And not being happy even with all this, everyone was always ready to jump ship for more money and toys.

The final straw was the attitude, "I must work from home; you people are distracting me and I do much better work at home."

Well, to quote John Lennon, "The dream is over."

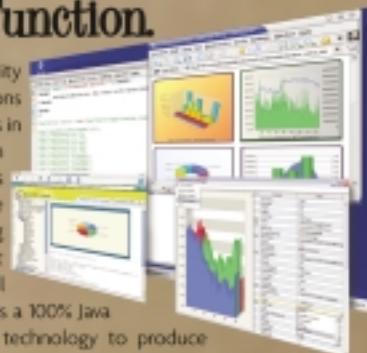
There is no question that outsourcing is bad for America. I look at this every day, editing America's Job Market ([americas-job-market.com](http://americas-job-market.com)). Quarterly driven corporate greed perpetuates the practice. If things continue in the direction they are currently going, corporate America will someday have to begin outsourcing customers for their products. ☺

# A Fresh Brew of Chart FX...



## With a Robust Blend of Form and Function.

10 years of creating quality developer charting solutions gives us an edge over others in the industry. So, although we're the new kid on this block, we've been in the neighborhood for a long time. The maturity of Chart FX and our support will prove it. Chart FX for Java is a 100% Java component that uses JSP technology to produce charts in a variety of formats: PNG, JPEG, SVG and FLASH. Developed using JDK 1.4, it supports J2EE 1.3 and J2SE 1.4. Chart FX for Java is available as a Server-side Bean and an Enterprise Java Bean (EJB) that runs on most popular Java Application Servers, with features like borders, gradients, alpha blending, anti-aliasing and transparency.



Additionally, it will produce native Windows output in the form of a ActiveX or .NET component. Easy data population from JDBC, XML, Text Files, API and other popular data sources. Includes a Designer, which is a stand-alone tool to set the visual attributes of the chart, and a Resource Center containing the Programmers Guide, the Javadoc API and hundreds of samples. For more information or to download a trial version, visit [www.softwarefx.com](http://www.softwarefx.com).



# Chart FX

for JAVA

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

In the US: (800) 392-4278 • In the UK: +44 (0) 117 905 8733 • In Germany: 0800-24 27 839



As Java development evolves, so does JProbe®

## JProbe®

Find the cause of J2EE code performance, memory and threading problems faster than ever before with Quest JProbe. New investigative features for finding memory problems combined with dramatic performance improvements mean even quicker resolution of problems in your application, servlet, JSP and EJB code.

JProbe Suite

JProbe Profiler

JProbe Memory Debugger

JProbe Threadalyzer

JProbe Coverage

Part of the Quest Performance Management Suite for the J2EE Platform



For more info and a free eval, visit:

<http://www.quest.com/jdj>