

JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

October 2003 Volume:8 Issue:10

www.JavaDevelopersJournal.com

International Conference & Expo

Edge 2004

EAST Feb. 24-26, 2004
Boston, MA

details on pg. 45

From the Editor
Java the Brand,
Not the Technology
Alan Williamson pg. 5

j2EE Insight
The Kids Are Alright
Joseph Ottinger pg. 8

j2SE Insight
I Love Logging!
Jason Bell pg. 32

j2ME Insight
Quality Is Job n?
Glen Cordrey pg. 50

Industry News
pg. 64

RETAILERS PLEASE DISPLAY
UNTIL DECEMBER 31, 2003

\$5.99US \$6.99CAN



SYS-CON
MEDIA



by Rich Helton • page 40

-  **Viewpoint: It's Not Over Till the Fat Client Sings** *Java on the desktop*  **Joe Winchester** **6**
-  **Load Balancing: Easing the Pain of Enterprise Application Deployment** *Increasing productivity is just the beginning* **Jeff Browning** **10**
-  **Feature: Zip Objects, Zap Wait Time** *Compress data - decrease your network traffic*   **Robert Beckett** **20**
-  **Forum: Java Games Development** *Part 3 of 3 - Developing 3D games and more*  **Jason R. Briggs** **34**
-  **Feature: The Location API** *Simplify access to mobile positioning methods*  **Sven Haiges** **52**
-  **DelegatorFactory: Multiple Inheritance in Java** *Of diamonds and dynamic proxies*  **Dan Haywood** **58**
-  **JSR Watch: From Within the Java Community Process Program** *From 'Tiger' to portlets to MIDPs*  **Onno Kluyt** **64**
-  **From the Inside: One IDE to Rule Them All** *To capture mind share, Java needs a killer IDE* **Henry Roswell** **66**

Parasoft Corporation

www.parasoft.com/jdj10



Why do industry leaders choose Zero G?

Because, Zero G is the industry leader in multi-platform software deployment and updating. But, it didn't just happen overnight. Since 1996 we have listened, adapted, and built our award-winning InstallAnywhere® and PowerUpdate® products based on constant interaction with our customer-partners. That's why they have become the developers' deployment solution of choice worldwide. It's no wonder that industry leaders like Sun Microsystems, Novell and Borland choose us.



Your software deployment partner

www.ZeroG.com



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.



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



Maximized



Alan Williamson
Editor-in-Chief



Java the Brand, Not the Technology

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 (ICP)
- Alan Williamson (SYS-CON)
- Joe Winchester (IBM)
- Blair Wyman (IBM)

Editorial

- Editor-in-Chief: **Alan Williamson**
- Executive Editor: **Nancy Valentine**
- J2EE Editor: **Joe Ottinger**
- J2ME Editor: **Glen Cordrey**
- J2SE Editor: **Jason Bell**
- Contributing Editor: **Jason R. Briggs**
- Contributing Editor: **Ajit Sagar**
- Founding Editor: **Sean Rhody**

Production

- Production Consultant: **Jim Morgan**
- Associate Art Director: **Louis F. Cuffari**
- Associate Editors: **Jamie Matusow**, **Gail Schultz**, **Jean Cassidy**, **Jennifer Van Winckel**
- Online Editor: **Lin Goetz**
- Technical Editor: **Bahadir Karuv, PhD**

Writers in This Issue

- Robert Beckett, Jason Bell, Jason R. Briggs, Jeff Browning, Glen Cordrey, Sven Haiges, Dan Haywood, Rich Helton, Onno Kluyt, Joseph Ottinger, Alan Williamson, Joe Winchester

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. 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: 201 782-9600

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 © 2003 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, carrie@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
Frank Cipolla: 845 731-3832, 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.



Like many of you, I keep an eye on what Sun is doing as a company. I keep an eye on their press releases, official statements, and general product lines. I don't necessarily pay a great deal of attention to the output unless it specifically mentions Java because, as we know, there is more to Sun than just Java.

I've known about their "Mad Hatter" (Linux desktop) project for some time now. It's essentially a collection of open source projects, all designed to work together in one desktop, running initially on Linux. This is Sun's continued play to become a single-stop solutions company. As Scott McNealy remarked at his Sun Network 03-Q3 keynote in San Francisco in September: "We're the IT company, not you."

However, at the Sun Network, Sun announced it was renaming the project the "Java Desktop System." Excuse me? The Java Desktop System? What's that all about? Deep within the bowels of Sun, someone has decided that associating it with Java in a clear concise marketing message will increase the success of this project. Maybe it will, maybe it won't. But it's one heck of a gamble.

Leaving aside the fact that the name is misleading since the "Desktop System" has very little to do with Java, I am prompted to ask why not call it the "Star Office Desktop System" or the "Mozilla Desktop System" – or even the "Sun Desktop System"? When you say "Java Desktop System," I instantly think of the ill-fated JavaStation (remember the Java shark fin terminals?). This was a Java desktop. Everything running was Java. The only native app you could run was a Java class file. That's how native it got.

As announced, the Java Desktop System on the other hand is not a pure Java platform. It's not a single JVM controlling the whole desktop. Java is merely the recommended language; Sun is encouraging us to write our

apps destined for that desktop in Java. To call it the Java Desktop System is being disingenuous.

I wish Sun lots of success with this, but historically they aren't renowned for succeeding in the software world. Sun ONE didn't rock any boats, Forte didn't shake any trees, and we can only hope that Project Rave is going to come within at least a sniff of all the marketing hype.

The issue is that the Java brand has been hijacked for a project that has very little to do with Java. The press are already writing about how the Java Desktop System is aimed to compete with Microsoft Windows and how well the one integrates with the other.

My fear for the wider Java community of developers is this: If this fails to knock Microsoft out of the desktop space, guess what will be blamed by the critics and the analysts? Not Sun, but Java. Sun is playing a game of Russian roulette with the prize china! Naturally Sun legally owns the Java brand, so you can argue that they can do with it whatever they please. But surely they have a duty of care to the community, a responsibility to it. Yes they own it, but aren't they more a custodian of Java?

If this blows up in Sun's face, it blows up in all our faces. Java is struggling on the desktop as it is and only now are we clawing back with a strong, viable solution that can offer a serious alternative (see Joe Winchester's **Viewpoint** on page 6). Surely we want to avoid doing anything that is going to set this momentum back.

We in the wider Java community ought not to allow Sun to take risks like this with "our" brand. We have too much invested in Java for Sun to be misusing the Java name without either the Java community or Java having a major play in it.

If the Java Desktop System fails, it won't be because of the Java component. My question is: If this doesn't work out, how will we all be able to convince the corporate world of that? ☺

Alan Williamson, when not answering your e-mails and working on the next issue of *JDJ*, heads up a small team dubbed the "Thunderbirds of the Java industry," providing on- and off-site rescue for Java projects in trouble. For more information visit www.javaSOS.com. You can also read his blog: <http://alan.blog-city.com>.

alan@sys-con.com

It's Not Over Till the Fat Client Sings



Joe Winchester

Reports of Java's death on the desktop may be somewhat premature. A recent Giga group report, "Return of the Rich Clients", predicts that in the next three years browser-rich clients will grow by 350%, stand-alone clients by 250%, while HTML will decline by 50%. Two major facts are contributing to this change: problems associated with traditional client development being solved and HTML not providing a powerful enough user interface for all GUI requirements. Both of these are good news for Java.

For stand-alone clients, Java has advanced on several fronts recently. The J2SE team delivered substantial performance improvements to Swing in 1.4.2, as well as a great Windows XP and GTK look and feel. Meanwhile the Eclipse project created SWT that uses a rich set of cross-platform native controls over and above those provided by AWT. Newsgroup flame wars often pitch the two as rival GUI toolkits; however, hopefully this will become a thing of the past as the current interoperability problems are tackled.

One of the problems associated with traditional client/server development is a systems management issue of how to ensure that the software at all end points is kept up-to-date. HTML largely avoids this by creating the page marking up the client UI each time a request is made to the Web server. Arguably, this on-demand preparation of the GUI is the single largest reason HTML has become such a ubiquitous programming model. Java Web Start, however, solves the original distribution problem by using the Web as a mechanism to deliver a traditional Java application to the client. Each time the program is run it checks against the Web server to see whether a newer version is available and, if required, downloads the updated JAR files. JWS programs run within Java's security model; however, client-side caching and the use of local JRE avoid the issues that plagued applets.

Joe Winchester

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

winchest@uk.ibm.com

Several Java hybrid clients also exist that run Java on the server, but instead of delivering HTML to the browser, they use plug-ins to create a richer UI experience. With the ultra-lightweight client from Canoo, a J2EE programmer uses Swing peer classes as if writing client-side Java, requests are marshaled back and forth as XML, and a full Swing GUI is actually created on the client. The RSWT SourceForge project does the same except it uses SWT as its Java toolkit. Other examples of Java hybrid technologies are classic blend, droplets, and thinlets, all of which deliver a rich GUI to the user through a Java server-side programming model.

It's not going to be easy for Java to win back the client as it faces stiff competition from Microsoft with Windows Forms, and Visual Basic as the incumbent client development language.

With this level of activity in the client Java space, **Java Developer's Journal** is launching a new section entitled "Desktop Java." This will include solid technical content to help you understand more about the various projects and technologies, as well as editorials and news. The mistakes have been made, the lessons learned, and Java is now well positioned to recapture some of its lost pride as a GUI platform. We hope you enjoy the new section. ☺

References

1. "Return of the Rich Clients" report available to registered Giga customers: www.gigaweb.com
2. SWT/Swing: www.chronline.com/currentnews/b096965887/e6c9a380256d940018c964
3. Java Web Start: <http://java.sun.com/j2se/1.4.2/docs/guide/jws/developersguide/contents.html>
4. ULC: www.canoo.com/ulc
5. RSWT: <http://rswt.sourceforge.net>
6. Classic blend: www.appliedreasoning.com/products/what_is_Classic_Blend.htm
7. Droplets: www.droplets.com
8. Thinlets: www.thinlet.com

**SYS-CON
MEDIA**

President and CEO:

Fuat Kircaali fuat@sys-con.com

Vice President, Business Development:

Grisha Davida grisha@sys-con.com

Group Publisher:

Jeremy Geelan jeremy@sys-con.com

Advertising

Senior Vice President, Sales and Marketing:

Carmen Gonzalez carmen@sys-con.com

Vice President, Sales and Marketing:

Miles Silverman miles@sys-con.com

Advertising Sales Director:

Robyn Forma robyn@sys-con.com

Director, Sales and Marketing:

Megan Ring megan@sys-con.com

Advertising Sales Manager:

Alisa Catalano alisa@sys-con.com

Associate Sales Managers:

Carrie Gebert carrie@sys-con.com

Kristin Kuhnle kristen@sys-con.com

Editorial

Executive Editor:

Nancy Valentine nancy@sys-con.com

Associate Editors:

Jamie Matusow jamie@sys-con.com

Gail Schultz gail@sys-con.com

Jean Cassidy jean@sys-con.com

Jennifer Van Winckel jennifer@sys-con.com

Online Editor:

Lin Goetz lin@sys-con.com

Production

Production Consultant:

Jim Morgan jim@sys-con.com

Lead Designer:

Louis F. Cuffari louis@sys-con.com

Art Director:

Alex Botero alex@sys-con.com

Associate Art Director:

Richard Silverberg richards@sys-con.com

Assistant Art Director:

Tami Beatty tami@sys-con.com

Web Services

Vice President, Information Systems:

Robert Diamond robert@sys-con.com

Web Designers:

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

Accounting

Accounts Receivable:

Kerri Von Achen kerri@sys-con.com

Financial Analyst:

Joan LaRose joan@sys-con.com

Accounts Payable:

Betty White betty@sys-con.com

SYS-CON Events

President, SYS-CON Events:

Grisha Davida grisha@sys-con.com

Conference Manager:

Michael Lynch michael@sys-con.com

National Sales Manager:

Sean Raman raman@sys-con.com

Customer Relations

Circulation Service Coordinators:

Niki Panagopoulos niki@sys-con.com

Shelia Dickerson shelia@sys-con.com

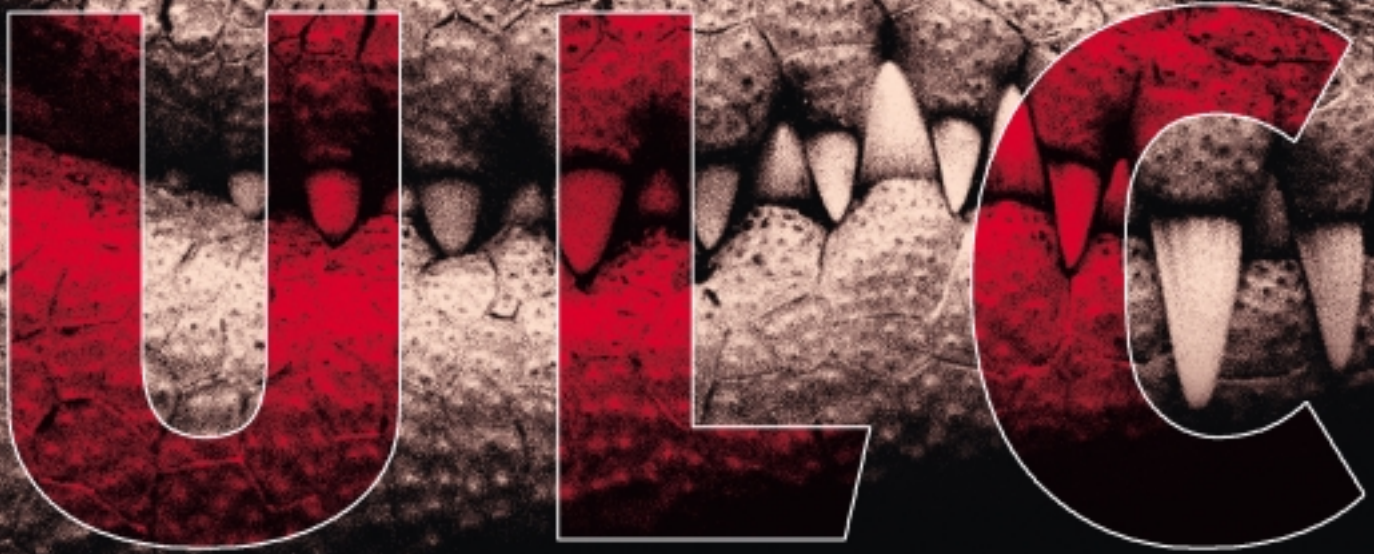
Edna Earle Russell edna@sys-con.com

IDJ Store Manager:

Rachel McGouran rachel@sys-con.com

NEW!
Release 5.1
out now!

Java thin clients made easy



ULC

Ultra Light Client with full GUI capability

easy development

- no worries about client-server code split
- no technology mix: pure server-side Java development
- widgets transparently split between client and server
- extensible Swing-based widget set incl. source code

easy deployment and operation

- application deployment on server only
- lean and generic display engine on client
- optimized for low-bandwidth communication
- seamless J2EE integration (Web/EJB container)
- Web integration (applet, Java Web Start)

canoo.com

Canoo Engineering AG Tel +41 61 228 94 44
Kirschgartenstrasse 7 Fax +41 61 228 94 49
CH-4051 Basel <http://www.canoo.com/ulc/>
Switzerland ulc-info@canoo.com

Download your free trial today!



Joseph Ottinger

J2EE Editor



The Kids Are Alright

Every month we're told again and again how Java is on its way out. A multibillion-dollar company tells us that, while hiring other large companies to say the same thing. One sad group of souls says it's because of Java's licensing, or the lack of features available in other languages or frameworks, and another wails that Java's too flexible, while another set says that Java's too slow.

It ain't so.

The oddest thing about all this, to me, is that Java is one of the best languages around, and all the yammering about it is an indication of its health. You don't eulogize for very long; you only discuss healing while the patient's still alive, while there's hope. Microsoft doesn't bother advertising how much better Windows is than XENIX or OS/9 or OS/2 – those are battles it's won. Instead, Microsoft reserves its fire for Linux and, dare I say it, Java.

There are a lot of obstacles in front of Java developers today. Most of them are philosophical in nature, and can be fixed by a little bit of logical thought and education; I find that in my own sphere of expertise, developers tend to be unaware of many critical aspects of J2EE programming, mistaking some of the component specifications as being representative of the entire J2EE spectrum (which I'm guilty of myself!), or simply riding on the assumptions offered to them by other similarly inexperienced coders. Some of the obstacles are simply based on outdated information (such as the "Java is slow" myth); others are based on advertising. Some of the issues are bullet-point related (such as the "Java needs templates" discussions), and even more are based on simple defeatism.

All this is fine, really. I don't mind that these issues and others even exist, because we're aware of them. Sun Tzu, in "The Art of War," said that you

should know your enemy; these are our enemies. There are those who pick on this column and various Web sites for being negative when Java doesn't need it – they want positive analyses and happy success stories only; I think these Pollyannas need to wake up and smell the Java! You can't solve a problem you don't know about, and while I try to be constructive in nature, even those who are less constructive serve a very valuable purpose: they show you things that need to be improved. Even if the things they say are wrong, their data had to come from somewhere – perhaps that's an opportunity to improve the documentation or education process.

In fact, I'd go so far as to say that I distrust a purely glowing review. I've been really happy with a few products, to the point where I'm happy to agree that their warts are very minor – Borland's Optimizeit ServerTrace is one of these products. That doesn't mean the warts aren't there or can't be improved, and I offer feedback in an attempt to make every product everything it can be.

Java's no exception. In my last editorial, I asked where the components were (Vol. 8, issue 9), and I'm glad to say there were a number of vendors and programmers offering answers. I've been willing to critique Sun's management of Java along with many others, and here's the thing that shows Java's health: those critiques have been acknowledged and answered!

Sometimes the answer isn't what we want to hear, but the fact is that little of this truly falls on deaf ears.

Java's alright, and unless things drastically change, the continued growth of Java will go on, unstemmed. It's still fun, still useful, still worthwhile – and will be for a long, long time. ☺



The Kids Are Alright

Every month we're told again and again how Java is on its way out. A multibillion-dollar company tells us that, while hiring other large companies to say the same thing. One sad group of souls says it's because of Java's licensing, or the lack of features available in other languages or frameworks, and another wails that Java's too flexible, while another set says that Java's too slow. It ain't so.

Easing the Pain of Enterprise Application Deployment

To provide the best application performance, reliability, scalability, and security for J2EE applications, many large organizations utilize network load-balancing appliances and application switches. However, coordinating the deployment of applications between application developers and network managers can be a slow, painstaking process. An opportunity now exists to help automate the process of deploying and updating Java applications to save development time, ease frustrations of deployment, and more.

Joseph Ottinger is a consultant with Fusion Alliance (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 JDJ Editorial Advisory Board.

josephottinger@sys-con.com

More little black lines than ever before.



* new or updated features with Crystal Reports 9 Advanced

With over 50 new features inside Crystal Reports 9, it's impossible to list all the finer points of our high-productivity reporting solution.

More features. More productivity. More flexibility.

Allowing detailed control of end user report viewing and runtime interaction, Crystal Reports is an ideal solution for developers seeking efficient integration of dynamic application reporting and presentation layers using Java, .NET, and COM.

Crystal Reports is also a fundamental component of Crystal Enterprise, a web-based information delivery system. This reliable, scalable, and secure technology is an excellent starting point for application developers who need a platform that can grow.

With over 14 million licenses shipped, and chosen by industry-leading technology partners like Microsoft and SAP for its high-productivity solutions, Crystal Decisions is a trusted software vendor.

Find out more. Visit our web site for free online seminars, the Developer Zone, demos, and evaluation software at: www.crystaldecisions.com/lbl/.

To buy now, call 1-800-877-2340, visit our online store at www.crystaldecisions.com/lbl/estore/, or contact your reseller.

© 2003 Crystal Decisions, Inc. All rights reserved. Crystal Decisions, Crystal Reports, Crystal Enterprise, and the Crystal Decisions logo are either a trademark or registered trademark of Crystal Decisions, Inc. in the U.S. and/or other countries. All other trademarks or registered trademarks referenced are the property of their respective owners.

Easing the Pain of Enterprise Application Deployment

Increasing productivity is just the beginning



Jeff Browning

To provide the best application performance, reliability, scalability, and security for J2EE applications, many large organizations utilize network load-balancing appliances and application switches. However, coordinating the deployment of applications between application developers and network managers can be a slow, painstaking process for companies choosing the many advantages of the network-based, load-balancing switches.

By developing an application to manage the time-consuming coordination between these two teams, developers can self-serve their needs more quickly, network managers can save valuable time previously spent performing manual tasks, and companies can experience more flexible application deployment and updates.

Budget-conscious IT departments can also save enormous amounts of scarce financial resources and time. On a network basis alone, conservative estimates show that companies can save nearly \$20,000 per network engineer per year in time spent managing this process by building a very basic application deployment-and-update application. At the same time, the client experience is improved through better orchestrated application deployment.

Summarily, an opportunity now exists to help automate the process of deploying and updating Java applications in order to save development time, ease frustrations of deployment, and gain flexibility to deploy new applications on demand while offering the best client experience.

The Current State of Enterprise Applications

It should come as little surprise to developers that as Web applications have flourished, so have demands on

the application servers required to deliver them. The days of deploying an application to one server and hoping it delivers performance and reliability are a thing of the past.

As a result, applications are typically deployed in a redundant fashion across multiple identical application servers. This ensures that the application is available when requested, even if one application server is out of service or experiencing technical difficulties. Traditional "load-balancing" solutions help virtualize a specific application IP address so that when people access a URL or IP address, the load balancer determines which redundant server is best suited to accept and respond to the application request.

The advantages of load balancing are significant. Most serious Web sites and e-business applications today employ some form of load balancing because it ensures a predictable client experience. Load-balancing solutions also help increase application performance by distributing requests to the best performing servers. In some cases, they can deliver more cost-effective applications due to lower server license costs and server expenditures.

When it comes to application reliability and load-balancing options, companies usually consider two alternatives for increasing the reliability and performance of their applications. One choice is software load balancing, which is included as a feature with most Java application servers. In this scenario, application servers help distribute load by utilizing some form of clustering or "round robin" load balancing. While this capability is usually included in the price of application servers, it comes at a cost to server performance. In many cases, software load balancing can inflict a 20-40% performance hit to each application

server running the load-balancing software.

Alternatively, companies choose hardware solutions that help provide load balancing and more intelligent IP traffic management to realize significant advantages while complementing their existing software clustering. A key advantage to this is extremely high reliability (99.999% or less than 5.3 minutes of downtime per year). For applications utilizing HTTPS for security, these devices can also offer performance optimization such as hardware-based SSL processing and acceleration. These devices can also provide more intelligent balancing schemes for complex applications using persistence. In the event a server goes down, the user session can typically be routed to the next available server for seamless, transparent application continuity.

While network devices provide advantages for high availability, reliability, and performance perspective, there's an added benefit for application developers and network staff who are responsible for deploying and updating applications: these devices provide an ideal control point to help orchestrate application updates and deployment at the ingress and egress point of all application requests.

Application Deployment and Update Challenges in the Enterprise

Deploying or updating application components is a topic of pain and frustration for many companies. While they desire the high availability, reliability, performance, and security that the combination of network load balancers and application servers provides, orchestrating application deployment requires collaboration between the application and network teams.

Updates require access to servers to

Borland Software Corporation

go.borland.com/j6

perform application updates and enhancements on a regular basis and usually require manual intervention by network administrators. Typically, this process requires application teams to request manual assistance from the network team to stop servers for updates and then restart them when updates are completed.

This approach has proven to be a burden for both the network and application teams. It takes longer to complete updates. There is an increased chance of human error and deploying new applications takes too long. To further complicate these matters, update times must be scheduled during periods experiencing the least amount of application traffic – commonly 2:00 a.m. on a Saturday morning. In almost all cases, the manual network procedures can be automated to shorten application development life cycles, enhance developer productivity, and roll out applications instantly for the best client experience.

A contributing factor to this challenge is the limited information provided by the J2EE specification. The J2EE spec, which vendors rely upon to build application servers and developer tools, provides guidance for installation and configuration. But it neglects to address execution, which is left to the product provider's platform. While many application server vendors have developed their own update and deployment capabilities, they are usually proprietary in nature. When combined with network devices in true enterprise environments, the combination of limited specifications and vendor-specific solutions can further complicate matters.

Finding the Solution

An opportunity exists to build intelligent applications that bridge the gap between the J2EE applications, servers, and the network that supports them.

When considering application development options, the two most likely approaches include SNMP and Web services based on SOAP/XML. While both offer the ability to help automate functions necessary for deploying and updating applications, SNMP has inherent security issues. For instance, the only access control available for SNMP is the ability to restrict access on IP addresses and alter the public string. Further, communications are done via clear text, which can enable other applications to spoof addresses and requests. Finally, SNMP requires developers to learn specialized

MIB vocabularies to develop applications.

To me, Web services via SOAP/XML appears to be a much more viable option. Based on widespread industry support, it offers rapid development with many different development environments and enables the use of virtually any language to fit developer needs. Therefore, I'll utilize SOAP for the following application example.

Application Overview

An application can be created quickly to help automate the update process. Applications can be full-featured, Web-client solutions that enable developers to log in and control which servers or applications they wish to update. Or, this same functionality can be combined with existing code management solutions to streamline the deployment process.

For this example, I'll outline the basic building blocks that can work for either scenario. To gracefully update application components, the update application must locate the server resources supporting an application, determine which servers need to be updated, gracefully remove existing traffic and connections to the server hosting the application, and enable the update process to begin. After the application components are updated, initialized, and tested, the application must also gracefully reintroduce the new applications to client requests.

Key Terms and Definitions

While network load-balancing technologies are not new, some of the terms and function definitions may not be familiar to application developers. Since these terms relate to the interfaces and methods I will be discussing in this example, here are some basic definitions of terms common to network devices:

- **Node:** A specific combination of an IP address and port (service) number associated with a server in the array that is managed by a network load balancer.
- **Pool:** Composed of a group of network devices (called members). The network load balancer distributes requests to the nodes within a pool based on the load-balancing method and persistence method you choose when you create the pool or edit its properties.
- **Member:** A reference to a node when it's included in a particular pool. Pools typically include multiple

member nodes.

- **Virtual server:** A specific combination of a virtual address and a virtual port associated with an application or Web service managed by a network load balancer or other type of host server.
- **Connection:** An active TCP connection to a specific server.

Application Description and Code Samples

For this example, the functions are relatively generic and represent logical steps needed to deploy or update application components on servers managed by a network load-balancing device. The code samples, designed to provide guidance on how to build your own application, are specific to F5's iControl API. Directions for how to learn more about this API and obtain more samples are provided at the end of this article. For other networking vendors, you can visit their respective Web sites for additional information on how to use the APIs they make available to developers.

Query for Pool Associated with Virtual Server Address

To locate the pool of nodes or members associated with a virtual server, query the virtual server address (see Listing 1). The response will indicate which pool(s) are available to manipulate. In many cases, the virtual server is the primary URL or IP address associated with a particular Web service or application.

Query Pool for Members

To determine which members (servers) are available for application updates, query the pool associated with the virtual server (see Listing 2). From the response, you'll have a listing from which to choose. This will be the total collection of servers that you will have access to when deploying new applications or updating existing applications.

Determine How Many or Which Member(s) to Target

In many cases, this is a business-process decision. The number or quantity of members you will want to target for updates comes down to a simple question: How many servers can you afford to take out of service at any given time? Your decision may be based on a number of factors including current server load, server resources provisioned for various peak times, or possibly something as elegant as calculating the average number of persistent appli-



FINALLY, BUSINESS **SOLUTIONS THAT** **WORK WITH EXISTING** **TECHNOLOGIES** **AND NONEXISTENT** **BUDGETS.**

You need to get more out of what you have. We have just the thing: solutions based on our open technology platform, SAP NetWeaver™. Because it's preconfigured to work with your current IT investments – and it's fully operable with .NET, J2EE and IBM WebSphere – SAP NetWeaver reduces the need for custom integration. That lowers your total cost of ownership for your entire IT landscape and gets you quicker ROI. Everything a CIO wants (and a CFO didn't think was possible). Visit sap.com/netweaver or call 800 880 1727 for details.

THE BEST-RUN BUSINESSES RUN SAP



Infragistics, Inc.

www.infragistics.com

Infragistics, Inc.

www.infragistics.com

cation connections.

One option, although not defined in the article, could include the creation of server groups to be utilized as “update groups.” These could then enable switching between groups when updates are executed (see Listing 3).

Set State to Disable Targeted Members

Once the targeted members (servers) have been identified, the next step is to proceed with setting the state of the members you wish to update. This step effectively prepares the servers by ensuring that no new client connections can be established. Further, it ensures that no new traffic is sent to the servers. Once this state is set, new connections and traffic will be sent to other members in the pool, ensuring that new requests always get an active server and not a server going through the update process.

```
void setNodeState(IPPortDefinition[] members, int state) {
    ...
    // Setting method parameters
    soapParams.addElement(new
    Parameter("node_defs",
        IPPortDefinition[].class, members,
        null));
    soapParams.addElement(new
    Parameter("state", Integer.class,
        state, null));
    // Set Call object method and parameters
    call.setMethodName("set_state");
    ...
}
```

Establish When to Drop All Connections from a Member

Once the state has been set to disable targeted members, the decision must ultimately be made as to when all remaining connections will be disabled. There's a variety of criteria from which to choose depending on your business needs. A good approach can include determining the threshold of the tolerable number of connections to drop. For example, once only two connections remain, drop them and disable. Another alternative is using the time since the member was disabled. A further option could include looking for connections from specific client IP addresses. When these connections cease, disable the server.

Once the member is disabled and all connections are dropped, it's considered offline and ready for updates.

```
void setNodeAvailability(IPPortDefinition[] members, int avail) {
```

```
...
// Setting method parameters
soapParams.addElement(new
Parameter("node_defs",
    IPPortDefinition[].class, members,
    null));
soapParams.addElement(new
Parameter("state", Integer.class,
    avail, null));
// Set Call object method and parameters
call.setMethodName("set_availability");
...
}
```

Update Application Components on the Target Server

At this point, the developer has many choices as to the best way to distribute the application. Most simply use existing component distribution techniques that can include FTP, programmatic deployment, or even integration with an existing code management system.

Once the update process has been completed, appropriate steps should be taken to initialize or register new components on the server. It's further recommended that new component testing be performed prior to restoring the member to an active pool.

Update Application Monitor

One feature usually deployed with intelligent network load-balancing solutions is application health checking. Monitors check application availability at periodic intervals based on specified criteria. This ensures that if an application goes down, no new client requests are routed in its direction.

If an application monitor needs to be created or has been in use for previous versions, now is the time to create the new monitor. Once created, it is associated with the newly updated member and the old monitor is deleted (see Listing 4).

Set Updated Member to Receive Connections

Now that application updates are completed, the server tests out correctly, and the monitor is set, the member is ready to be reintroduced to the pool of servers supporting an application. The member state needs to be set to receive connections and placed in an enabled mode.

```
void setNodeAvailability(IPPortDefinition[] members, int avail) {
    ...
    // Setting method parameters
    soapParams.addElement(new
```

```
Parameter("node_defs",
    IPPortDefinition[].class, members,
    null));
    soapParams.addElement(new
    Parameter("state", Integer.class,
        avail, null));
    // Set Call object method and parameters
    call.setMethodName("set_availability");
    ...
}
```

Set State to Allow New Traffic

Once enabled, the member is reintroduced to the pool and can begin allowing new client requests that will respond with the new application just deployed.

```
void setNodeState(IPPortDefinition[] members, int state) {
    ...
    // Setting method parameters
    soapParams.addElement(new
    Parameter("node_defs",
        IPPortDefinition[].class, members,
        null));
    soapParams.addElement(new
    Parameter("state", Integer.class,
        state, null));
    // Set Call object method and parameters
    call.setMethodName("set_state");
    ...
}
```

Putting It All Together

At this point, there are a variety of options for how a member should be reintroduced to the incoming flow of requests. In most cases, how it is reintroduced depends on factors based on overall methodology and the process outlined for spot or rolling updates.

For instance, an organization may want to schedule regular updates when applications or sites have specific availability requirements. Alternatively, this application could enable an organization to update applications at any time, given that now there's control of how the application is deployed, such that clients will continue to have seamless application access throughout the process.

One deciding factor may include whether the application is a basic application or a more complex application using session persistence. For basic applications, an application based on the building blocks provided may stage blocks for updates. For example, in a pool of six servers, the first three may be updated and upon being reintroduced into the pool, the remaining three are removed to follow the same update process.

Diagnose and resolve J2EE performance problems ...now in production

PerformaSure™

Now you can take advantage of PerformaSure's exclusive Tag and Follow technology to diagnose and resolve J2EE performance problems in production. Tag and Follow traces and reconstructs live end-user transactions across the JVMs, web/application servers and databases of your distributed J2EE system. Get to root cause with PerformaSure.

PerformaSure's new features reduce overhead to rock-bottom levels:

- Ultra low-overhead production-grade agents
- Component-level instrumentation and detail dial
- Automatic sampling and filtering, and more



Take a tour of PerformaSure today:

<http://java.quest.com/performasure/jdj>

For applications where persistence is a factor – particularly long persistent sessions – a deployment strategy might include creating a new pool for the updated servers, adding the servers to that pool, and then remotely telling the network device to begin directing new client requests to the newly formed pool of updated servers. The benefit to this approach is that all remaining persistent sessions running on servers using the old application version can time out in their own graceful manner. This helps ensure that client experiences go smoothly. Only after the session is complete will the servers be cycled through the update process.

Application Benefits

By deploying this solution to ease application deployments and updates in a J2EE environment, companies are able to increase developer productivity, reduce deployment frustration, reduce management overhead, and gain flexibility to deploy new applications on demand. With this solution, the customer can:

- **Reduce time spent managing the deployment of applications:** Some customers have reported savings of 3–5 hours per week per network administrator. Given only four network administrators saving six hours per week, this could equate to a savings of 144 workdays/year. Given the cost of network management, this can relate to significant cost savings for any organization.
- **Accelerate deployment of applications through automation:** Replacing manual processes with automation enables faster updates and deployments and thus increases the developer time available to continually improve applications. Further, it provides end users with access to more current applications – increasing their net productivity.
- **Reduce errors associated with manual intervention:** Some estimates suggest that nearly 80% of network configuration errors can be attributed to manual, human intervention. By automating this process, you can reduce a significant volume of configuration problems that limit mission-critical application availability.

Related Information

For additional information and sample code to build similar applications, please visit <http://devcentral.f5.com>.

Jeff Browning is a product manager for Seattle-based F5, a provider of secure application traffic management products.
j.browning@f5.com

Listing 1

```
String getPoolName(String addr, long port) {
    Response resp;
    Vector soapParams = new Vector();
    IPPortDefinition vs = new IPPortDefinition();
    vs.setAddress(addr);
    vs.setPort(port);

    // Setting method parameters
    soapParams.addElement(new Parameter("virtual_server",
        IPPortDefinition.class, vs, null));

    // Set Call object method and parameters
    call.setMethodName("get_pool");
    call.setParams(soapParams);

    // Set Object URI and method name
    call.setTargetObjectURI(urnVS);
    // invoking SOAP method call
    resp = call.invoke(destURI, urnVS);

    // Checking response for status of service request
    if (resp.generatedFault()) {
        // Request failed, retrieving SOAP fault object
        Fault fault = resp.getFault();
        System.out.println("Fail to process set_state operation");
        System.out.println("Fault Code: " + fault.getFaultCode() +
            " Fault String: " + fault.getFaultString());
    } else {
        poolName = (String) resp.getReturnValue().getValue();
    }
    return poolName;
}
```

Listing 2

```
IPPortDefinition[] getMemberList(String poolName) {
    Response resp;
    Vector soapParams = new Vector();
    IPPortDefinition[] nodes = new IPPortDefinition[1];
    // Setting method parameters
    soapParams.addElement(new Parameter("pool_name", String.class,
        poolName, null));

    // Set Call object method and parameters
    call.setMethodName("get_member_list");
    ...
    nodes = (IPPortDefinition[]) resp.getReturnValue().getValue();
    return nodes;
}
```

Listing 3

```
long getCurrentConnections(String nodeAddr, long nodePort) {
    ...
    node.setAddress(nodeAddr);
    node.setPort(nodePort);
    soapParams.addElement(new Parameter("node_def",
        IPPortDefinition.class, node, null));
    ...
    call.setMethodName("get_statistics");
    ...
    NodeStatistics stats =
        (NodeStatistics) resp.getReturnValue().getValue();
    currentConnections =
        stats.getConnection_stats().getCurrent_connections();
}
```

Listing 4

```
void createMonitorTemplate(String name, String template,
    int type, long timeout, long interval) {
    ...
    MonitorIPPort monIPP = new MonitorIPPort();
    monIPP.setIpport(member);
    monIPP.setAddress_type(type);

    CommonAttributes comAttr = new CommonAttributes();
    comAttr.setTimeout(timeout);
    comAttr.setInterval(interval);
    comAttr.setDest_ipport(monIPP);
    comAttr.setParent_template(template);

    // Setting method parameters
    soapParams.addElement(new Parameter("template_name",
        String.class, tempName, null));
    soapParams.addElement(new Parameter("template_type",
        Integer.class, tempType, null));
    soapParams.addElement(new Parameter("template_attributes",
        CommonAttributes.class, comAttr, null));

    // Set Call object method and parameters
    call.setMethodName("create_template");
    ...
}

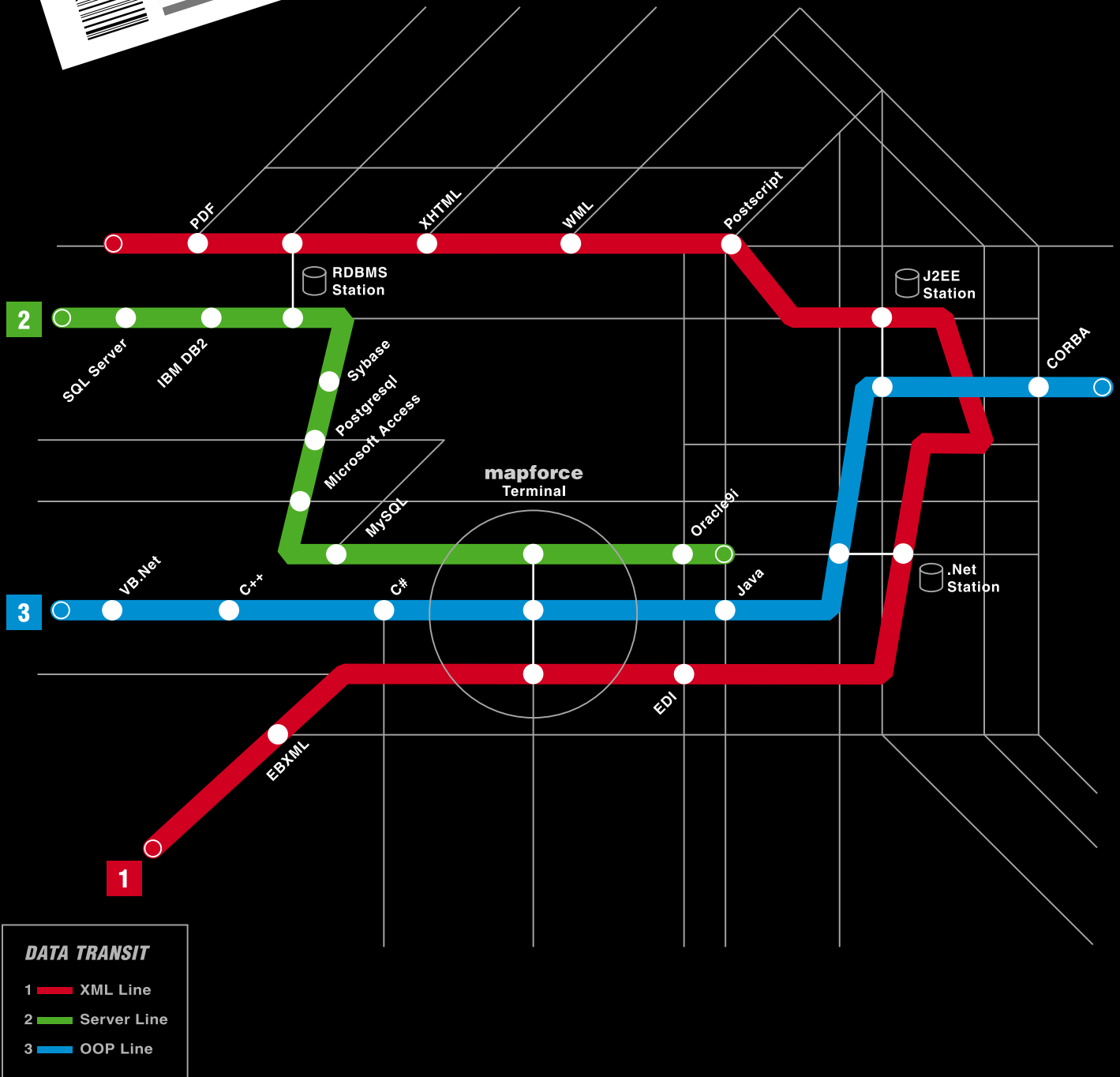
void createNodeAssoc(String[] tempNames, String node, long port) {
    ...
    MonitorIPPort mon = new MonitorIPPort();
    mon.setIpport(member);
    mon.setAddress_type(type);

    MonitorAssociation monAssoc = new MonitorAssociation();
    monAssoc.setNode_definition(mon);
    monAssoc.setTemplate_names(tempNames);

    // Setting method parameters
    soapParams.addElement(new Parameter("monitor_association",
        MonitorAssociation.class, monAssoc, null));
    ...
    call.setMethodName("create_association");
    ...
}
```



Next Stop, Mapforce 2004!



Introducing Mapforce 2004! A powerful new XML mapping tool from Altova, producer of the industry standard XML Development Environment, XMLSPY. Mapforce is a visual data mapping tool, which auto-generates custom mapping code in multiple output languages such as XSLT and Java, to enable programmatic XML-to-XML or XML-to-Database data transformations. Mapforce 2004 is the hassle-free transit for moving data from one format to another, enabling efficient information reuse. **Download a free trial now!**

ALTOVA®

www.altova.com

All other trademarks are the property of their respective owners.



J2ME



J2SE



J2EE



HOME

ZIP OBJECTS, ZAP WAIT TIME



DECREASE YOUR NETWORK TRAFFIC

BY ROBERT BECKETT

As the capabilities of our distributed applications increased, so did our consumption of bandwidth. In 1998, our server sent objects no larger than 50K to a group of users on a local network. By 2002, we were passing an average of 500K per object, with some as large as 1.5MB.

More important, the distribution of our user base grew from 50 to over 1,500, with some users based across the country from the server. Add in a group of users roaming on their modem connections and the full scale of our bandwidth issues become clear. We were presented with a problem faced by many developers of distributed systems: reduce bandwidth usage and client wait time without removing any functionality. This article shares our solution to this problem, providing you with the simple code that helped us eliminate over 80% of our network traffic.

Evaluating bandwidth is quite simple. The developer has two options: get more of it or use less of it. Given the magnitude and expense of expanding bandwidth on a nationally distributed application, it was clear we had to find ways to reduce the amount of bandwidth required by our systems. It's important to note the wording: reduce the bandwidth usage, not the amount of data passed over the network. To preserve the functionality of the systems, we needed all the data being passed over the line. In the end, there was one conclusion: the data needed to be compressed.

As I researched compression in Java, I was looking for a way to pass in an object and receive a compressed object back. I found that there are a number of ways to compress sockets or build zip files on the disk, but not the object-level solution I was seeking. We needed an API that could be selectively implemented and used for the largest data objects and most critical applications without impacting other parts of the system. We also wanted the ability to compress an object one time, and use that same object for multiple downloads to client machines, essentially caching a compressed object.

During this research, I found an article on compression on the Java developer's Web site that laid out all the pieces to our solution (see Resources section). Using just a few of the classes in the `java.io` and `java.util.zip` packages, we were able to build an API to compress any serializable Java object. Being the kind of developer who prefers simplicity, I was excited at the ease of use and performance of the underlying Java classes as well as the API we built. We were able to develop and integrate our solution in just under two days, resulting in more than an 80% reduction in network traffic and astounding improvements in client wait times.

A Compression Factory for Serialized Objects

The Java compression functions are located in the `java.util.zip` package, where the `Deflater` class compresses byte arrays and the `Inflater` class decompresses byte arrays. As you may have noted, both of these classes perform compression routines on byte arrays. Therefore, to compress an object, the first step is to translate it into a representation of bytes, which begins with the `Serializable` interface.

When an object implements the `Serializable` interface, it can be represented as a stream of bytes. This byte stream can be written using the `ObjectOutputStream.writeObject()` method and reconstituted using the `ObjectInputStream.readObject()` method, allowing for a simple translation of a byte stream to and from an object. This ability to serialize an object, capturing the resulting byte stream into a byte array, provides a usable input for the compression methods available in the `java.util.zip` classes.

Using this approach, we will accept a serialized object, write the object into a byte array, and then compress the array. The array of compressed bytes, along with a few other key variables, will be stored in a new object, `cZipObject`, which is shown in its entirety in Listing 1. The `cZipObject` will encapsulate the compressed version of the input object. The

IBM Rational

ibm.com/rational/seamless



cZipObject can then be serialized to transfer across the network. On the receiving end, the byte array will be extracted from the cZipObject, decompressed, input to a byte stream, and then reconstituted into an object. This process is not truly compressing the object, but compressing the serialized representation of the object and its data.

To easily integrate these compression routines on both the server and client side, we'll create a cZipFactory class that will contain all the methods for compressing and decompressing objects. We'll create a number of methods along the way that can be of direct use, such as a byte compression method. By encapsulating both the compress and decompress functions into a single class, we can add the functionality to both the client and server by creating a single object. This will allow us to compress objects sent from the server to the client as well as from the client back up to the server.

The first step is to convert the Serializable object into a byte array. This can be achieved by using the ObjectOutputStream with an underlying ByteArrayOutputStream from the java.io package. First, we'll create a new ByteArrayOutputStream that will capture the byte stream when the object is written. We'll then create a new ObjectOutputStream, write the serialized object, and then extract a byte array from the ByteArrayOutputStream.

```
try {
    ByteArrayOutputStream byteOut = new ByteArrayOutputStream();
    ObjectOutputStream objOut = new ObjectOutputStream(byteOut);
    objOut.writeObject(inObj);
    byte[] dataArray = byteOut.toByteArray();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

With this code, we now have the ability to translate any object that implements the Serializable interface into a byte array capable of compression. The resulting byte array contains the details of the object as well as the object's data. The array contains the essential structural and data attributes to replicate the object and all its content. The next step is to compress the data contained in the byte array, thereby compressing the serialized representation of the object.

There are a few simple steps to compressing byte arrays using the Deflater class from the java.util.zip package. First, we'll create a new array for the compressed bytes. Without a method to accurately predict or estimate the size of the byte array resulting from compression, it's advisable to create an array of equal size to the noncompressed bytes and then shrink the array once the compression is complete and the true size can be determined.

The next step is to create a new instance of the Deflater class, passing in the desired compression level in the constructor. There are a few options for compression level, each with benefits and drawbacks. The best compression option provides the greatest reduction in byte size at the expense of increased processing time. The best speed option provides a good compression level, usually 80% or better, in the shortest possible time. I usually opt for best compression, finding the extra milliseconds in processing time worth the decreased object size. For more information on the available compression levels, refer to the JavaDocs for java.util.zip.Deflater.

Once the Deflater object has been created, call the setInput(byte[]) method providing the byte array we extracted from the object serialization. Invoke the finish() method to inform the Deflater class that all inputs have been defined. Next, call the deflate(byte[]) method, providing the byte array

to house the compressed data. When this method completes its execution, the data has been compressed and populated in the output byte array. The getTotalOut() method in the Deflater class will return the total number of bytes that were written in the output byte array. Using the new array size, we'll create a byte array to the exact size of the compressed output. We'll then use the System.arraycopy function to copy the bytes from the temporary array into the exact size array.

For ease of use, we'll encapsulate these steps into a single method named CompressBytes in the cZipFactory object (see Listing 2). Now, when we need to compress a byte array, we can invoke a single method:

```
byte[] bytesCompress = ZipFactory.CompressBytes(DataArray);
```

There are two key pieces of data required to quickly and accurately decompress the object: the byte array containing the compressed data and the original size of the serialized byte array. When the byte array is decompressed, it will be written into another byte array. Knowing the size of the decompressed array will not only make the decompression more efficient, it will also ensure accuracy. To save the byte array and original size easily, we will encapsulate them in a new instance of the cZipObject class.

```
cZipObject cZipObj = new cZipObject();
cZipObj.setData(bytesCompress, origSize);
```

By combining all these steps, we can now create a method that accepts any Serializable object and returns a cZipObject. This is the Compress method in the cZipFactory class, shown in its entirety in Listing 3. Using the new method in cZipFactory greatly simplifies the integration of object compression functions. First, we create an instance of the cZipFactory class, providing the desired compression level during object creation.

```
cZipFactory ZipFactory = new
cZipFactory(java.util.zip.Deflater.BEST_COMPRESSION);
```

Using the new cZipFactory class, we can compress a serializable object using a single line of code:

```
cZipObject newZObject = ZipFactory.Compress(inObj);
```

When the client or receiving machine obtains the cZipObject, it needs to be decompressed and reconstituted into an object. To achieve this, we'll create another method in cZipFactory to handle the Decompress operation. This method will extract the byte array from the provided cZipObject, decompress the array, and then translate the bytes into an object. The Decompress method in cZipFactory will return a Serializable object, which can be cast into the original type of object.

Using the java.util.zip.Inflater class, we can easily decompress the byte array in a few lines of code. Given the compressed byte array and the original size of the byte array, the Inflater class can be used to decompress the byte array. As this function could be useful in a variety of situations, we'll create a method in the cZipFactory class named DecompressBytes. The method will accept a byte array containing the compressed bytes and a primitive integer for the size of the decompressed array. At this point, it's very important that we know the original size of the byte array (see Listing 4). Without this information, it wouldn't be possible to accurately predict the total size of the decompressed bytes

Parasoft Corporation

www.parasoft.com/jdj10

Calculating the Benefits of Compression

There are a number of benefits to using serialized object compression, most notably the reduction in the size of the serialized output. The performance gain is directly related to the average object size, the bandwidth of the client connections, and the CPU processing power of the server and client machines. When determining whether to implement a compression function, these factors should be projected in order to ensure a positive gain. Consider this simple equation to determine if compression routines would be beneficial:

$$[(\text{Object Size bytes} \div 8) \div [\text{Line Speed kbs}]] = \text{Avg. Download Time (ms)}$$

$$[10000 \div 8] \div 128 = 625 \text{ ms}$$

Now, reduce the average object size by 80% and recalculate the download time; this time add an additional 100 milliseconds for processing time.

$$[(\text{Object Size bytes} \div 8 \div 0.2)] \div [\text{Line Speed kbs}] + 100 = \text{Compress Download Time (ms)}$$

$$\{[(10000 \div 8) \div 0.2] \div 128\} + 100 = 225 \text{ ms}$$

In the chart in Figure 1 we see how the slight increase in processing time required for compression can create tremendous gains in download time.

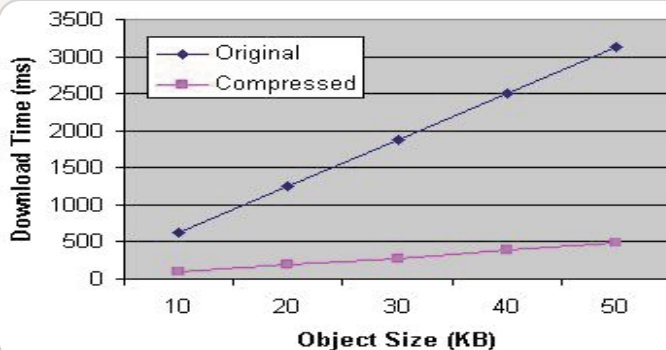


Figure 1 Impact of compression on download time

Regardless of the bandwidth from the server to the client, compression routines will have a definitive impact on network usage (see Figure 2).

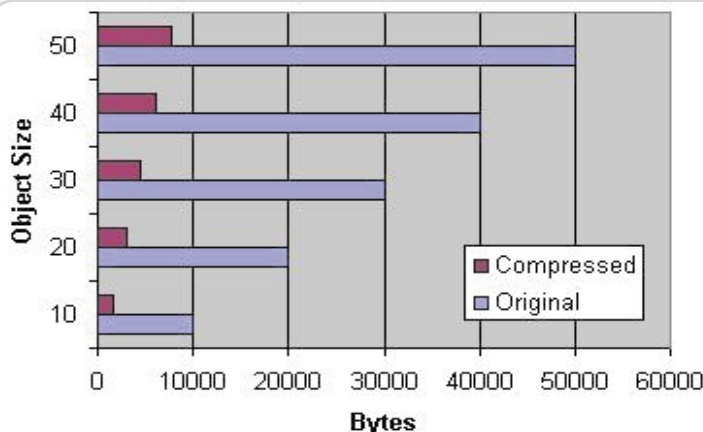


Figure 2 Object size reduction from compression

It's important to remember that at some point the law of diminishing

without extracting the data in a loop. Knowing the original size of the byte array makes the decompression code easier and more efficient.

With the ability to decompress a byte array in place, we then move to the process of converting the bytes back into a usable object using an instance of `ObjectInputStream`. First, we'll create a `ByteArrayInputStream` using the decompress byte array. Using the byte stream, we'll construct a new `ObjectInputStream` to reconstitute the object. By invoking the `readObject` method, the `ObjectInputStream` will translate the byte stream into a usable object. To simplify our coding, we'll place this code in a method named `ConvertByteToObject` in the `cZipFactory` class (see Listing 5).

The final step is to create a `Decompress` method in the `cZipFactory` class that will accept a `cZipObject` and return a `Serializable` object. The completed `Decompress` method is shown in the `cZipFactory` class in Listing 3.

Using the `cZipFactory` class, we can now decompress a `cZipObject` using a single line of code:

```
Serializable retObject = ZipFactory.Decompress(newZipObject);
```

The `Serializable` object can then be cast into its original form or in the same line of code as the call to `Decompress`:

```
Vector vClientList = (Vector)ZipFactory.Decompress(newZipObject);
```

In the end, the `cZipFactory` provides easy-to-use methods that translate serializable objects to and from compressed representations of objects. The entire compression API can be quickly implemented in just a few lines of code. Another important feature is the ability to use the function selectively rather than a system-wide change, such as compressing a socket. The resulting `cZipObject` can be extended or expanded to meet the requirements of an application or can be treated like any other Java object. This also allows for the reuse of a `cZipObject`, allowing the developer to cache a compressed object, effectively eliminating the need to redundantly perform compressions.

A Simple Client List Example

Now that we've built the classes to compress `Serializable` objects, we'll work through an example using the new objects. To begin, let's create a vector of client names. For our example, we'll create a vector with generic content, but you could imagine this list of clients being derived from a database call, an XML document, or some other data source.

```
Vector vClients = new Vector(1000);
for (int i = 0; i < 1000; i++)
    vClients.add("Client # " + i);
```

The resulting vector, `vClients`, contains 1,000 entries and when serialized is 14,046 bytes. If the client machine connects using a 28.8 modem, they will retrieve this vector at approximately 3.33 KBS. At this throughput rate, it'll take the client machine approximately 4,200 milliseconds to download this list of 1,000 clients. If we wanted to add in compression, we'd add this line of code on the server:



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>

Features	Basic	Professional	Enterprise	Dedicated
Monthly Cost	us\$9.00	us\$17.00	us\$39.00	us\$191.00
Servlet Spec	Latest	Latest	Latest	Latest
JSP Spec	Latest	Latest	Latest	Latest
EJB Spec			Latest	Latest
Private JVM	x	x	x	x
Database	1	1	5	Unlimited
e-Commerce		x	x	x
Tier 1 Center	x	x	x	x
Accounts/Server	120	120	35	1
RAM/Server	2GB	2GB	2GB	256MB
Max Memory Heap	20 MB	30 MB	60 MB	256MB
RAID Diskspace	60MB	200MB	900MB	40GB
Bandwidth/Month	3GB	10GB	20GB	55GB
Backup	x	x	x	x
Domains	1	2	5	Unlimited
Email Accounts	2	20	100	Unlimited
Servlet Contexts	1	4	20	Unlimited
Control Panel	x	x	x	x
FTP	x	x	x	x
Telnet/SSH		x	x	x
Web Mail	x	x	x	x
Web Stats		x	x	x
Perl/PHP		x	x + ASP.NET	x + ASP.NET
Dedicated IP			x	x
Engine Choices	x	x	x	x
WAR/EAR	x	x	x	x
Tomcat		Latest	Latest	Latest
JBoss			Latest	Latest
Jetty	Latest	Latest	Latest	Latest



J2ME



J2SE



J2EE



HOME

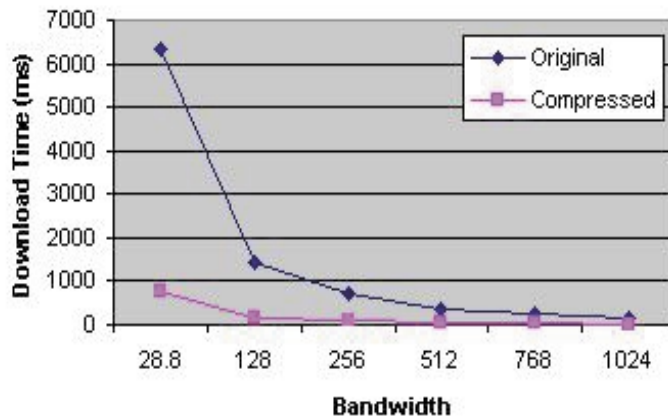


Figure 3 Diminishing returns on compression of 20K object

returns becomes prevalent. For example, if the average size of the object before compression is 5,000 bytes, then compression could reduce this to as little as 1,000 bytes. The total expense of this compression would be about 100 milliseconds. If the client machines were on 28.8 modems, the compression would have a positive impact, reducing client wait time by about 1,100 milliseconds. However, if the client machines were on 512K connections, downloading the original 5,000 bytes would only take about 90 milliseconds. Even though the 1,000 bytes would take 17 milliseconds, we have now added additional processing time for the compress and decompress operations, potentially creating a negative return, and not significantly impacting download time.

The chart in Figure 3 helps to illustrate how the benefits of compression on client wait time can be quickly reduced in higher bandwidth environments. It's important to note that while client wait time may not be significantly reduced by compression, network traffic will always be reduced. Even though the end user may not notice improvements, the network will always benefit from the reduction in throughput.

```
//Using a pre-existing cZipFactory class instance
cZipObject zoClients = ZipFactory.Compress(vClients);
```

On the client machine, we add this line of code to decompress the cZipObject:

```
//Using a pre-existing cZipFactory class instance
Vector vClients = (Vector)ZipFactory.Decompress(zoClients);
```

Using this example, the Compress method executes in approximately 40 milliseconds. We would then transmit the zoClients object to the client machine, which when serialized is 2,296 bytes. At 28.8 modem speed, the cZipObject instance is downloaded to the client in approximately 690 milliseconds. The client then decompresses the cZipObject, casting the contents into a vector. The Decompression operation on the client takes an additional 30 milliseconds. The total time using compression was 40 + 690 + 30 = 760 milliseconds.

When compared to the original download time of 4,200 milliseconds, the compression technique saved 3,440 milliseconds of client wait time and reduced the total object size by 11,750 bytes, resulting in 83.6% less bandwidth consumption. This is more than five times faster and is achieved with a few simple lines of code on the server and client.

Listing 6 provides a simple testing class that was used for this example and the benchmarks quoted in this article. By using this simple testing class, you can see that when applied to larger data structures, the compression functions make a

more profound impact on bandwidth reduction and client wait times.

Expense of Compression

There are two primary expenses to this compression technique: increased memory usage and CPU cycles. This approach is compressing the serialized representation of an object, which requires that the object be serialized into an array that's then compressed and included in another serializable object. In addition to the increase in memory usage, there will be an increase in CPU utilization. The compression routines are comprised of arithmetic operations, which will result in increased CPU usage during deflation and inflation processing. For larger installations of these compression routines, it would be reasonable to expect notable increases in server CPU usage,

which would need to be analyzed in terms of frequency and the size of the objects being compressed. As a benchmark, in one installation the server processed approximately 10,000 compressions an hour on objects ranging from 10K to 350K. The addition of compression functions resulted in approximately a 3% increase in CPU usage.

Another important factor to remember is that the client machines will also have increases in memory usage and CPU utilization to decompress the objects, or compress objects being sent to the server. The speed of these decompression routines will depend on the client machine hardware.

Conclusion

If you are writing distributed Java applications, whether they're EJB systems or custom RMI solutions, the introduction of compression routines can provide tremendous improvements to the response time and bandwidth consumption of your programs. One of the primary advantages to the approach presented here is its simplicity, allowing the developer to continually work with objects and avoid the compression functions. Using the cZipFactory also allows the developer to avoid socket-level operations or the creation of disk files, retaining the structure of existing programs and making it possible to selectively implement the functions. Another benefit of the cZipFactory is the use of standard Java libraries, making the compression function available in both J2SE and J2EE applications.

For our applications, the performance of the compression routines has been excellent, with minimal server impact and network usage down by 85%. Today, of the approximately 3,000 client machines using the compression classes, there have been no reports of problems with CPU utilization or memory usage. Overall, the introduction of compression was the single largest performance improvement made in our five year development effort. ☺

Resources

- "Compressing and Decompressing Data Using Java APIs": <http://developer.java.sun.com/developer/technicalArticles/Programming/compression/>
- *Object Serialization in Java*: <http://java.sun.com/j2se/1.4.2/docs/guide/serialization/>
- *Java Documentation for java.util.zip package*: <http://java.sun.com/j2se/1.4.2/docs/api/java/util/zip/package-summary.html>
- *Java Documentation for java.io package*: <http://java.sun.com/j2se/1.4.2/docs/api/java/io/package-summary.html>



Robert Beckett is the chief architect for The Software Development Cooperative (www.thesdc.com), where he leads their efforts in building high-performance, scalable Java tools and solutions.

rbeckett@thesdc.com

The **only** Java components you need for J2EE or Swing development

JClass[®]

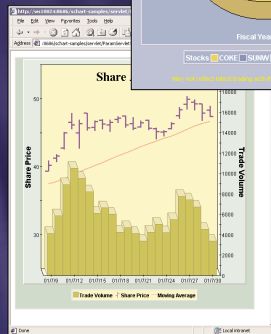
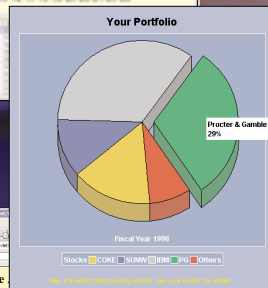
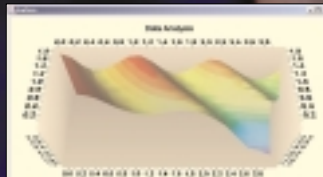
Rich client user interface and utility components.
Server-side web client interface and reporting
components. Whatever type of Java development
you're doing, JClass can help.

JClass ServerViews

Add professional content to your Servlet, JSP or J2EE
applications. Generate interactive charts with JClass
ServerChart and dynamic PDF
reports with JClass ServerReport.
Now fully XML and Web Services
ready!

JClass DesktopViews

Essential components for
client-side Java applications and
applets: 2D/3D charts, tables/grids,
data-entry fields, database access
and much more.



Evaluate and experience JClass today - visit:
<http://java.quest.com/jclass/jdj>



J2ME



J2SE



J2EE



HOME

Listing 1

```

1 public class cZipObject implements java.io.Serializable
2 {
3     private int iOriginalSize;
4     private byte[] compressData;
5
6     public cZipObject()
7     {
8         iOriginalSize = 0;
9     }
10
11    public synchronized void setData(byte[] inData, int
12        iOriginalSize)
13    {
14        compressData = inData;
15        iOriginalSize = iOriginalSize;
16    }
17    public synchronized byte[] getData()
18    {
19        return compressData;
20    }
21
22    public synchronized int getOriginalSize()
23    {
24        return iOriginalSize;
25    }
26 }

```

Listing 2

```

1 public byte[] CompressBytes(byte[] dataArray)
2 {
3     byte[] testBytes = new byte[dataArray.length];
4
5     cDeflate = new Deflater(iCompressLevel);
6     cDeflate.setInput(dataArray);

```

```

7     cDeflate.finish();
8     cDeflate.deflate(testBytes);
9
10    int iRetLen = cDeflate.getTotalOut();
11    byte[] retArray = new byte[iRetLen];
12    System.arraycopy(testBytes, 0, retArray, 0,
13        iRetLen);
14    return retArray;

```

Listing 3

```

1 import java.io.ByteArrayOutputStream;
2 import java.io.ByteArrayInputStream;
3 import java.io.ObjectInputStream;
4 import java.io.ObjectOutputStream;
5 import java.io.Serializable;
6 import java.util.zip.Deflater;
7 import java.util.zip.Inflater;
8
9 public class cZipFactory
10 {
11     private int iCompressLevel = Deflater.BEST_COMPRES
12         SION;
13     private Deflater cDeflate;
14     private Inflater cInflate;
15
16     public cZipFactory(int iCompLevel)
17     {
18         iCompressLevel = iCompLevel;
19     }
20
21     public cZipObject Compress(Serializable inObj)
22     {
23         cZipObject cZip = new cZipObject();
24
25         try {
26             ByteArrayOutputStream byteOut = new
27                 ByteArrayOutputStream();

```

ReportingEngines

JAVA REPORTING TOOLS FROM ACTUATE

REPORTS!



Formula One from ReportingEngines
Embedded reporting toolsets for J2EE application servers

When your users start screaming for reports, you should start reaching for Formula One from ReportingEngines.

Formula One provides embedded toolsets that enable Java developers to build simple to sophisticated reporting functionality into servlets, JSP and Java applications.

Formula One fully leverages the infrastructure of your J2EE application server without duplicating it with its own external server process. This allows you to schedule, secure, and deliver reports through

portals, email, or on demand as part of a Java application directly from WebLogic, WebSphere, JBoss, or any application server.

Design reports in a point-and-click environment from any data source, including XML, data streams and in-memory Java objects. Even drive reports programmatically with code and APIs.

See it today! Try it now! FREE!
 Visit www.ReportingEngines.com, see live demos, and download a free 30-day trial, tutorial, and training course.

30-Day Trial: www.ReportingEngines.com/info/trial3.jsp
888-884-8665 • sales@ReportingEngines.com

Copyright © 2003 ReportingEngines (a division of Actuate Corporation). All rights reserved. Formula One is a registered trademark of Actuate Corporation. Java and J2EE are trademarks and JSP is a trademark of Sun Microsystems, Inc., in the United States and other countries. All other trademarks are property of their respective owners. All specifications subject to change without notice.

Register Now!

bea.com/dev2devdays2003

or call

1-925-287-5156

1 DAY

20 CITIES

500 MINUTES OF TRAINING

1,000s OF DEVELOPERS

ALL THE CODE YOU CAN HANDLE

And not a single iota of hype.

BEA dev2dev Days is the worldwide, one-of-a-kind, no-nonsense seminar that gives you the tools you need to solve today's toughest development and integration challenges.

All in a single day. With no hype. And no spin. Just the facts.

You want code? We got code. Code-level demonstrations, in fact, on everything from architecting to building to integrating enterprise solutions on BEA's industry-leading WebLogic® 8.1 platform.

At the seminar, you'll learn information and techniques to help you build Web-based and service-oriented applications; work with XML and Web services; integrate with enterprise resources and applications; and much more.

But you have to act fast. A BEA dev2dev Days seminar is coming soon to a city near you, and spaces are limited. The cost to register is US\$169.00.

BEA dev2dev DAYS 2003

BY DEVELOPERS, FOR DEVELOPERS

Register Now!

bea.com/dev2devdays2003

or call

1-925-287-5156

SEPTEMBER – NOVEMBER 2003

COMING TO A CITY NEAR YOU:

Amsterdam
Atlanta
Bangalore
Beijing
Boston
Chicago
Dallas
Denver
London
Madrid
Mexico City
Munich
New York
Paris
San Francisco
Seoul
Stockholm
Tokyo
Toronto
Washington, D.C.

BROUGHT TO YOU BY:

dev2dev

attachmate

Confluent Software

FTP FAWCETTE
TECHNICAL
PUBLICATIONS

hp
invent

intel

SYS-CON
MEDIA

bea®



```

26      ObjectOutputStream objOut = new
27          ObjectOutputStream(byteOut);
28      objOut.writeObject(iObj);
29
29      byte[] dataArray = byteOut.toByteArray();
30      int origSize = dataArray.length;
31      cZip.setData(CompressBytes(dataArray),
32          origSize);
33      } catch (Exception e) {
34          System.out.println(e.getMessage());
35      }
36      return cZip;
37  }
38  public byte[] CompressBytes(byte[] array)
39  {
40      byte[] testBytes = new byte[array.length];
41
42      cDeflate = new Deflater(iCompressLevel);
43      cDeflate.setInput(array);
44      cDeflate.finish();
45      cDeflate.deflate(testBytes);
46
47      int iRetLen = cDeflate.getTotalOut();
48      byte[] retArray = new byte[iRetLen];
49      System.arraycopy(testBytes, 0, retArray, 0,
50          iRetLen);
51      return retArray;
52  }
53  public Object Decompress(cZipObj cZip)
54  {
55      byte[] unzipped = DecompressBytes(cZip.getData(),
56          cZip.getOriginalSize());
57      return ConvertByteToObject(unzipped);
58  }
59  public byte[] DecompressBytes(byte[] array, int
60      iLen)
61  {
62      byte[] retBytes = new byte[iLen];
63
64      try {
65          Inflater decompressor = new Inflater();
66          decompressor.setInput(array);
67          decompressor.inflate(retBytes);
68          decompressor.end();
69      } catch (Exception e) {
70          System.out.println(e.getMessage());
71      }
72      return retBytes;
73  }
74  public Object ConvertByteToObject(byte[] data)
75  {
76      Object objCache = null;
77
78      try {
79          ByteArrayInputStream fIn = new
80              ByteArrayInputStream(data);
81          ObjectInputStream objInput = new
82              ObjectInputStream(fIn);
83          objCache = objInput.readObject();
84          fIn.close();
85      } catch (Exception e) {
86          System.out.println(e.getMessage());
87      }
88      return objCache;
89  }

```

Listing 4

```

1      public byte[] DecompressBytes(byte[] array, int
2          iLen)
3      {
4          byte[] retBytes = new byte[iLen];
5
6          try {
7              Inflater decompressor = new Inflater();
8              decompressor.setInput(array);
9              decompressor.inflate(retBytes);

```

```

9          decompressor.end();
10         } catch (Exception e) {
11             System.out.println(e.getMessage());
12         }
13         return retBytes;
14     }

```

Listing 5

```

1      public Object ConvertByteToObject(byte[] data)
2      {
3          Object objCache = null;
4
5          try {
6              ByteArrayInputStream fIn = new
7                  ByteArrayInputStream(data);
8              ObjectInputStream objInput = new
9                  ObjectInputStream(fIn);
10             objCache = objInput.readObject();
11             fIn.close();
12         } catch (Exception e) {
13             System.out.println(e.getMessage());
14         }
15         return objCache;
16     }

```

Listing 6

```

1      import java.util.Vector;
2
3      public class cZipExamples
4      {
5          cZipFactory zipFactory;
6
7          static public void main(String[] args)
8          {
9              cZipExamples newZipExam = new cZipExamples();
10
11              newZipExam.ClientExample(10);
12              newZipExam.ClientExample(100);
13              newZipExam.ClientExample(1000);
14              newZipExam.ClientExample(10000);
15              newZipExam.ClientExample(100000);
16              newZipExam.ClientExample(1000000);
17          }
18
19          public cZipExamples()
20          {
21              zipFactory = new cZipFactory(java.util.zip.
22                  Deflater.BEST_COMPRESSION);
23          }
24
25          public void ClientExample(int iSize)
26          {
27              Vector inVector = new Vector(iSize);
28              for (int i = 0; i < iSize; i++)
29                  inVector.add("Client #" + i);
30
31              long lStart = System.currentTimeMillis();
32              cZipObj newZObj =
33                  zipFactory.Compress(inVector);
34              long lTime = System.currentTimeMillis() -
35                  lStart;
36
37              int iOrig = newZObj.getOriginalSize();
38              int iNew = newZObj.getData().length;
39
40              System.out.println("Zipped Vector object con-
41                  taining " + iSize + " from " + iOrig +
42                  " to " + iNew + " in time: " + lTime);
43
44              lStart = System.currentTimeMillis();
45              Vector vClients = (Vector)zipFactory.Decompress
46                  (newZObj);
47              lTime = System.currentTimeMillis() - lStart;
48
49              System.out.println("Returned Vector object in
50                  time: " + lTime);
51          }
52      }

```

Oak Grove Systems

www.oakgrovesystems.com/jdj



Jason Bell
J2SE Editor

I Love Logging!

A few months ago I wrote an editorial on the touchy subject of proper testing (Vol. 8, issue 6). Thanks to you there was much support (and a volume of information from Parasoft and how JTest linked with unit testing; this opened my eyes!). No one disagreed with me, but somehow I can't see unit-testing libraries being included with the next update of the software development kit.

This brings me to another touchy subject – proper logging. If you administer any type of Unix-like system, you'll be used to working with log files in some form or another. Ever thought of logging your Java programs like that? Okay, the majority of us have watched everything spew out of System.out.println() or printStackTrace() at some time or another. No matter what the class of application – high performance, low performance, high visibility, or just something on your own machine – it doesn't matter. You have to log your messages properly.

There are a number of logging packages out there. The main one is in the 1.4 SDK (have a look at the java.util.logging package), Log4J by the Apache Jakarta project, and then there are some lesser known loggers like LN2. All do the same thing: record information to make your life easier.

The reason people don't log info is pretty much the same reason people don't do proper testing: it's a mindset thing. The aim here is to create history to prove that your application is running well; it also helps you trace back errors and warnings. Watching your entire history hurl itself onto your console output is not real help to you or anyone else.

Archive your logs daily no matter how big they are. If you use or have used Apache, you may have come across a utility called rotatelog, which organizes your log files and resets the current one for a new day or if the log file size reaches a certain size. My logs are generated daily, then, when the month is up, I burn them to CD.

If there is any change of data, that transaction should be logged. You

need a user trail to see what is happening. It's not snooping; it's protecting yourself. What happens when someone claims an action happened when you know it didn't? You check the logs.

Dependency Rot

Another oversight is the maintenance of third-party Java libraries. With the amount of active project work on SourceForge, you should be updating these JAR files as much as you can. What I tend to see is a trail of old and unsupported libraries used in applications. What would happen if I updated to the most recent JAR file? Would the application still function? There's usually one way to find out.

A set of mirror sites that had all the up-to-date JAR files would be a good idea. Then we could update periodically. My favorite example of this is the apt-get program that's used in Debian Linux. It's one of the best updaters I've seen. The onus is still on the system administrator to update the package list first. When you run apt-get, it checks dependencies and related packages; you can also do first time installs with it.

Our responsibility does not stop once our applications are shipped to clients. We still provide a service, as we are a service-oriented industry. We have a duty to inform, update, and educate customers as we see fit and do it with a large amount of respect and politeness.

If we can get these two issues cracked, logging and dependency, we all stand a pretty good chance of extending our careers and making Java the platform it deserves to be – indispensable. ☺

References

- *Log4J*: <http://jakarta.apache.org/log4j>
- *LN2*: <http://enigmastation.com/ln2>
- *Debian*: <http://debian.org>
- *JTest*: www.parasoft.com/jsp/products/home.jsp?product=Jtest&itemId=11



Java and
Stream
Ciphers

40

I Love Logging!

If you administer any type of Unix-like system, you'll be used to working with log files in some form or another. Ever thought of logging your Java programs like that? Okay, the majority of us have watched everything spew out of System.out.println() or printStackTrace() at some time or another. No matter what the class of application – high performance, low performance, high visibility, or just something on your own machine – it doesn't matter. You have to log your messages properly.

Java Games Development

This final part of the three-part series concludes with a discussion on .NET, developing 3D games using Java, and whether the players have found any benefits to using different approaches to games development.

Jason Bell is the senior programmer for a B2B portal. He's also a keen supporter of people reading the API docs before asking questions. In his spare time he's involved with building RSS development tools.

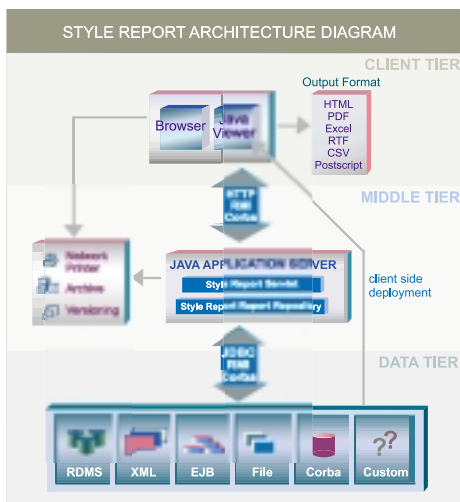
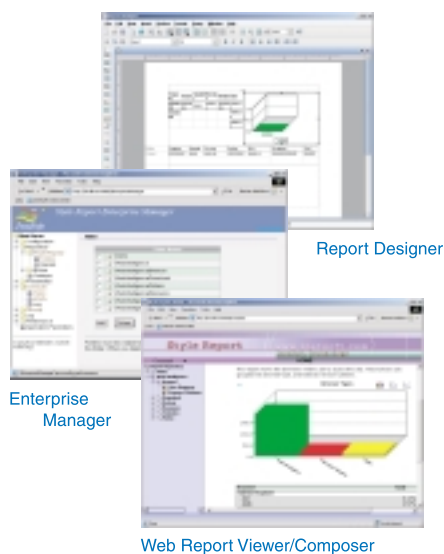
jasonbell@sys-con.com

Java & Web Reporting: with Java flexibility without Java complexity



Style Report: Information Vantage Point

Style Report blends the best of Java and Web to provide an award winning Web enterprise reporting solution. Based 100% on open standards such as XML, JSP, Java and Web services, Style Report allows you fully leverage existing IT infrastructure and common technical skills. One Report, Multiple Channel delivery and zero client ad-hoc reporting gives end users the freedom and frees IT from time consuming repetitive work. Designed for integration, Style Report is deployed and managed as a standard Web application or Corba/RMI service requiring not special skills. Download a free evaluation copy now!



Style Report Features

Flexible Visual Designer with JavaScript or VB Script	<input checked="" type="checkbox"/>
PDF, Excel, Word, CSV and other Export Format	<input checked="" type="checkbox"/>
Database/XML and other Datasources	<input checked="" type="checkbox"/>
Zero Client Browser Viewing	<input checked="" type="checkbox"/>
Built-in Drilldown/Parameters	<input checked="" type="checkbox"/>
Zero Client Ad-hoc Reports	<input checked="" type="checkbox"/>
Virtual Private Security Model	<input checked="" type="checkbox"/>
J2EE Drop-in Deployment	<input checked="" type="checkbox"/>
Advanced Scheduling	<input checked="" type="checkbox"/>
Report Versioning/Archive	<input checked="" type="checkbox"/>
Demand Paging/Report Cache	<input checked="" type="checkbox"/>
Portal/JSP integration	<input checked="" type="checkbox"/>

Java Games Development

Part 3 of 3

hosted by
Jason R. Briggs

Part 1 of this series appeared in the August issue of *Java Developer's Journal* (Vol. 8, issue 8), and Part 2 appeared in the September issue (Vol. 8, issue 9).

JD: Microsoft has received quite a lot of good press (or propaganda, however you'd like to look at it) for their .NET product, and there has even been discussion that it will be/is suitable for games development with good DirectX bindings. Since Java is only just out of the gate (in terms of commercial games development), do you see .NET providing serious competition in this burgeoning market, or are there some (perhaps hidden?) advantages to Java that might make the difference in this case?

Jeff K: Don't confuse .NET the platform with CLR the VM, or C# the language.

.NET will *not* be suitable for games. Its XML-based communications protocol is by definition slow and verbose, and that's before you get into the other architectural issues. I have no fear of .NET becoming the default back-end technology for networked games – it was designed with business apps in mind, which need totally different properties.

As for C# and CLR (which, as I understand it, are heavily based on MSVM), there are a number of points. MSVM has always supported DirectX through JDirect, but it hasn't taken off as a game platform. That they are continuing this support doesn't seem like anything new to me. C#/CLR has none of the benefits of Java (it's not cross platform, being YADC (yet another dialect of C) it's not likely to appreciably improve either productivity or code correctness.

About all it does is muddy the waters around Java, which in my opin-

The players are:

Jason R Briggs: *Java Developer's Journal* contributing editor and your host; games player when he has time, games developer...on occasion.

Gerardo Dada: Metrowerks' product manager for CodeWarrior Wireless Studio.

Erik Duijs: Former musician/engineer/producer with a (games) programming passion, now an IT consultant. Switched careers for the sake of better pay as well as maintaining a passion for music instead of "eating it" so to speak. Author of the Java Emulation Framework (JEF) and CottAGE.

Shawn Kendall: Developed Java and Java 3D-based game technology demos for Full Sail, Sun Microsystems, and I.M.I., and displayed at GDC and SIGGRAPH since 1999. He has five years of 3D technology teaching experience, and in 2002 founded Immediate Mode Interactive, LLC, a game technology company dedicated to the development and application of Java technology in computer games (www.imilabs.com).

Jeff Kesselman: Architect for game technologies, Advanced Software Technologies Group at Sun. He worked on the JDK performance tuning team and co-wrote *Java Platform Performance: Strategies and Tactics*.

Chris Melissinos: Sun's chief gaming officer and responsible for driving an industry-wide movement toward Java technology-based game development and building infrastructure programs for massively connected game play.

Caspian Rychlik-Prince: An IT consultant in the UK who for the last 10 years has specialized in client/server systems with RDBMS back ends. He has just released a new game, *Alien Flux* (www.puppygames.net).

Doug Twilleager: Chief architect of the Java Games Initiative at Sun Microsystems. One of the architects of Java 3D, he has also worked in the graphics research group at Sun looking at advanced rendering techniques and programmable shading.

David Yazel: VP of software development of trading systems and portfolio management systems at a leading financial investment company (by day), and by night a games developer for (and founder of) the Magicosm project (a 100% Java-based MMORPG).

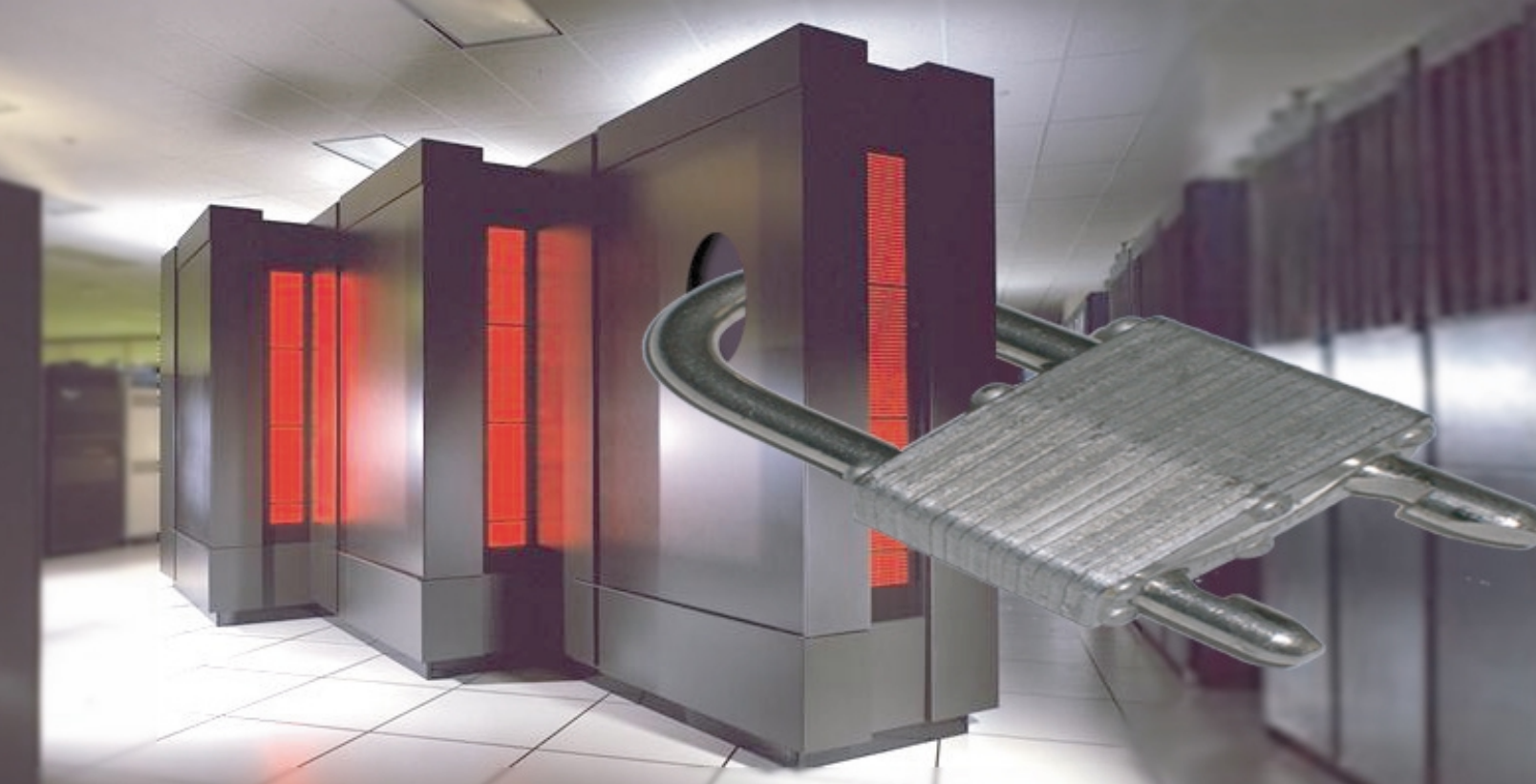
ion is maybe all it's supposed to do.

Today we have cross-platform technologies emerging in Java that provide all the benefits of DirectX without sacrificing portability, productivity, or code correctness. Examples of this are the LWJGL open source library project, and the open source bindings for Java to OpenGL (JOGL), OpenAL (JOAL), and controller input (JInput).

Doug T: As Jeff mentions, this mostly affects Java from the C# side of things. One advantage Java has is time. At GDC this year, there was a session on Managed DirectX, which is effectively DirectX for C#, and you could have replaced every instance of C# with Java in the presentation. We've been solving those problems for years, so we have a head start.

Cas P: [Microsoft's] VM technology is currently inferior, but not for long. The easy integration with a direct, ultra-capable gaming API (DirectX) means that it's only a matter of time before MS starts gaining a majority unless there's some direct competition in the same space from Sun. It's my belief that a three-pronged defense – for defense it is, as Java is the innocent blinking rabbit in Microsoft's headlights – is needed:

1. Gang together with game technologies, embrace and flourish – OpenGL and OpenAL are Sun's only hope against DirectX, so they'd better start supporting it, encouraging it, sponsoring it, and even helping it. If Sun is seen to be behind OpenGL to the point where their own products utterly depended on it, you'd get the positive feedback loop required to ensure that both support each other and grow.



unlock your business intelligence with ExtenBIS™

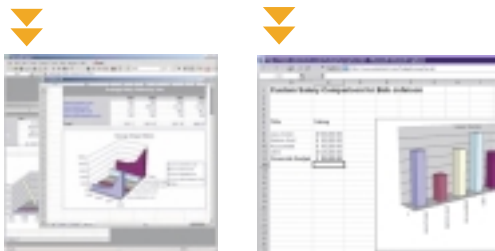


ExtenBIS™ Data Mapper

Extentech understands that your users need their data presented with all of the capabilities of an Excel™ spreadsheet, but don't have the bandwidth to learn yet another report design tool. Well, what better tool to create advanced spreadsheet reports with than Excel™ itself?

With the ExtenBIS™ Data Mapping and Deployment Wizard, you can visually map real-time database, session, & bean values to new or existing Excel™ templates with a drag-and-drop interface, then deploy it over the web to any server running the ExtenBIS™ Servlet with a single mouse-click. Your organization probably has hundreds of existing spreadsheet files with countless hours of labor locked up in business-critical formulas and formatting. By creating server-based Business Intelligence applications from these files, ExtenBIS™ is the key to achieving the maximum ROI from your existing BI content.

Utilizing a fully-customizable JSP-based portal [source-code included,] the ExtenBIS™ Business Intelligence Server delivers real-time Excel reports to your users anytime, anywhere.



Excel™ Spreadsheets

- ✓ Easy to use 3-Step Report Mapper & Deployment Wizard
- ✓ Secure Delivery of Reports by Web, Email, & File Drop
- ✓ Customizable Business Intelligence Portal with Source Code
- ✓ Customizable Business Intelligence Servlet with Source Code
- ✓ XML-based Report Definition Format
- ✓ Extensible Object-Relational CellBinder API with Source Code

EXTENBIS™
BUSINESS INTELLIGENCE SERVER

To download a free, fully functional evaluation copy of and for more information, please visit our web site at: <http://www.extentech.com/jdj>

Java is a Registered Trademark of Sun Microsystems Inc. Excel is a Registered Trademark of Microsoft Corporation.

2. Actively put major resources into the gaming space, gain the confidence of developers by being seen to do this, and start listening more keenly to suggestions and being a little more open about what's being done to address issues. I think we need a dedicated engineering, marketing, and support team for the whole issue, yet I feel that Jeff and Chris are the only Sun employees who are even trying, and they only seem to be let out of their cubicles once they've said their prayers to Solaris and flayed themselves for eight hours over a hot spreadsheet.
3. Hardware assault – the big, big question: Is there room for a Java-based console, and if not, why not? Perhaps Sun could even fund it by running it on a Sparc chip. If there's no room for a Java-based console, then truly Java is failing as the games platform of choice because it's the best place it could possibly be from a developer's perspective.

Erik D: I think the problem with .NET is that you are specifically choosing Microsoft and Microsoft alone and, in effect, you explicitly exclude a large market potential.

There's no way .NET will become available on a PlayStation, for example. Of course, we also have no Java (yet?) on PS, so currently development will stick to C++.



Java does have a longer history than .NET and has proven itself in many more ways than .NET has, such as the obvious platform independence. If Java did spread to PS3 and XBOX, I'd say "why C# and stick with PCs if we already have Java?" if I were a development company.

Shawn K: I do see serious competition from this because MS will do whatever it can to move developers to Windows,

games or otherwise. It doesn't matter if C#/.NET isn't exactly Java. They listen to their developers and changes get made. The fact that MS will make .NET work on the Xbox and future Xboxes makes it all the more compelling.

Also, MS makes games. They make one C# game that's a hit and it's a done deal. Developers and publishers will look at C#, and the same dynamic I stated before.

The only advantages for Java that I can see right now are:

1. Huge existing developer base and knowledge base.
2. Deep desire in game developers *not* to use MS stuff. That is a true phenomenon. Many working game developers I have talked to will try *anything* that is not MS tech.

I would like to point out some major disadvantages as well.

1. Licensing issues
2. Media APIs

JDJ: I know some of you are currently developing 3D games using Java. What was the reasoning behind your particular approach? Part 2 of this question is: Do you have any regrets? Are there any benefits to another approach that you wish you could take advantage of?

Cas P: First a quick mention about LWJGL. LWJGL is a modest library, still in development, and it aims to do very little other than to do things Java can't,

requests to enable one to create lots of windows and have them behave as they should. But then it wouldn't run on a console, would it? We've deliberately created a platform based around what we know the hardware and underlying O/S need to be for gaming: a display, some speakers, a few input devices – nothing more.

A side effect of LWJGL is it enables us to deploy games with less licensing restrictions: I can natively compile my games for Windows, which solves a bunch of other problems I won't go into here, and the demos come out under the magic 5MB download patience threshold.

I developed LWJGL in the first place because I find the OpenGL API easy and, most important, fast. Java 3D is so far above my head I barely know where to begin. This is the problem a lot of developers have with Java 3D. If you want to learn 3D programming, Java 3D isn't the thing for you.

The other reason for LWJGL was simplicity. It's a platform; there's a specification to program to. It's a bit vague right now because we're only in alpha, but it removes a lot of uncertainty and hassle. I discovered Alien Flux ran on Linux without any trouble at all. I was more surprised than anyone. Once it's up and running, it behaves in exactly the same way as the Windows one. You wouldn't know you were running Linux underneath it because it's full screen. I have the same hope for PlayStation 3

"I think the problem with .NET is that you are specifically choosing Microsoft and Microsoft alone and, in effect, you explicitly exclude a large market potential"

–Erik Duijs

like draw with OpenGL. There is a frequently misunderstood purpose to the library though. LWJGL is not an add-on library to J2SE to allow you to write OpenGL applications in Java. It was designed from the very beginning to be a portable console game library, even if it does end up being only completely theoretical. There's only one window in LWJGL, and we get a lot of requests for multiple windows. You can't even insist on a title bar and windowed mode; it's specifically meant to provide full screen-only games. We get a lot of

one day. You'd never know it was written for the PC because all the developer has to use is an empty black screen, some speakers, and three input devices, not all of which will even be present.

Another reason for LWJGL is performance – speed and size. I can write absolutely optimized code for LWJGL and it runs in hardly any memory. Alien Flux would even run on a PlayStation's 32MB of RAM with a little tweaking to the graphics. There are no caveats with regard to performance; all the knowl-

Borland Conference 2003

connect.borland.com/borcon03

edge about performance is from OpenGL, so it's very widely understood. If you write a slow LWJGL game, it's because you've used OpenGL incorrectly, not because we've written some dicky code in the library.

And no regrets. Not for one moment do I think I could have done it easier or better with any other technology. GL4Java, Direct3D, Java 3D – they're all deeply flawed when it comes to writing portable games in Java. My only real regret is that it's unlikely that we'll ever be endorsed by Sun. Although if I may make a prediction here: LWJGL is very likely to become the driving factor in Java games development.

Jeff K: In terms of using OGL incorrectly, that may be true for relatively simple games, but I'm not sure it carries into the A-lines games of today. OGL doesn't give you scene graph management; you're going to need to write that yourself. It doesn't give you physics; again you're going to have to code that. My guess is that you've covered about half of the core graphics issues with OGL, or about 5% of that 10% I talked about earlier.

For some folks that may be the perfect bar, for others it may be too low. However, all that being said, I do agree

like the idea of J3D completely hiding details; of course, this isn't fully justified because I haven't done much with J3D, but I do strongly believe you should always be able to get as low level as necessary where games development is concerned and I've seen people struggle with J3D issues that they don't seem to be in control of.

If J3D had been built on top of an OpenGL wrapper, I'd probably have used that because it obviously addresses the problem of LWJGL, that there aren't any scene graph implementations available for it yet.

Shawn K: One point I'd like to bring to light is the scene graph versus immediate mode argument. We have to remember that Java 3D is implementing a scene graph, and Java/OGL is simply immediate mode access through Java. There are pros and cons to both and they're completely independent of Java and Java 3D. That argument has waged on for generations.

We chose J3D for several reasons.

1. *Ease of use.* I never want to code immediate mode graphics if I can help it. It's like assembly for graphics. I use a high-level language and a high-level graphics system. I have also used scene graphs (from SGI to

being, when you know what you're doing, a scene graph or immediate mode are both great tools. When you don't know, a scene graph is a lot easier.

2. *Scene management.* Just like some LWJGL users are finding, eventually you have to build a scene graph anyway for any scene bigger than a bread basket.
3. *Common data structures for public as well as private interchange.* By this I mean, there's a thriving community of developers who code to J3D and easily trade huge code packages because they all use J3D classes and can be tested and integrated with very little modifications if written well. I'm not saying those actual packages will end up in a final production, but the integration-to-test phase is lightning fast. If all these developers were using OGL wrappers, the exchange would be close to nothing.

Look at the OpenGL community as it is. There's a lot of trading but it is all techniques and code "snippets," not loaders and behaviors or even collision systems because each OGL developer has his or her own scene graph equivalent structure on top of OGL that they will have to translate to for any techniques and code they get from the outside world.

“One problem, due to the high level and ease of use, is that developers think they can make great 3D graphics projects and not know *anything* about 3D graphics”

–Shawn Kendall

that being able to directly write to OGL is a good thing regardless of what you do on the top of it.

Erik D: I'm not married to LWJGL “till death do us part” or anything, but I do have a love affair with it because I like the low-level approach. It doesn't restrict me in any way and I believe it might become a key factor for well-performing games written in Java.

I've been playing a little with J3D, but I thought OpenGL was a lot easier to get into, and I like keeping in full control of things. Personally, I don't

PC) outside of Java and plan to keep on doing so. Eventually the class I teach adopted RenderWare (a pseudo-scene graph API) once it was an acceptable practice.

One problem, due to the high level and ease of use, is that developers think they can make great 3D graphics projects and not know *anything* about 3D graphics. Sometimes the questions that are posted on the J3D list are atrocious. OGL is so low level you have to have a bit better programming skill as well as more 3D knowledge to use it. I experience this in my courses as well. The point

4. *Better cross-platform than OGL alone.* We used Java/J3D for a long time as a true cross-platform solution. Our students' laptops ran best under DirectX, and our classroom workstations under OpenGL. For their work it was no problem to use either install and everything worked. This is less of an issue today as the laptops have better support for OGL and the workstations for DirectX. The current Java/OGL world is still an OGL on Linux/Solaris/Windows world so J3D has a slight edge there.
5. *Legacy.* It was the only way to do 3D in Java at the time we started. ☺



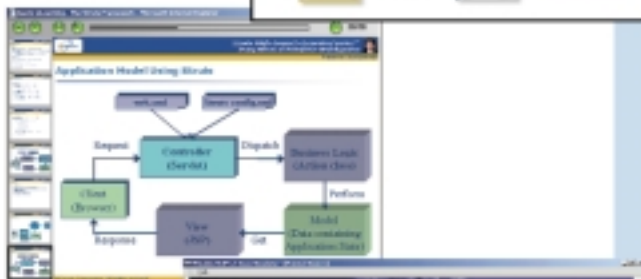
Jason R. Briggs is a Java programmer and development manager for a wireless technology company, based in Auckland, New Zealand. He is also a contributing editor of *Java Developer's Journal*.

jasonbriggs@sys-con.com



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.**



J2ME



J2SE



J2EE



HOME

In the 1990s, I worked extensively with the Winsock 2 interface and encryption when it first came out from Microsoft in Beta form; it was exciting in those early days of networking because it allowed you to easily encrypt data through the networks.

by Rich Helton

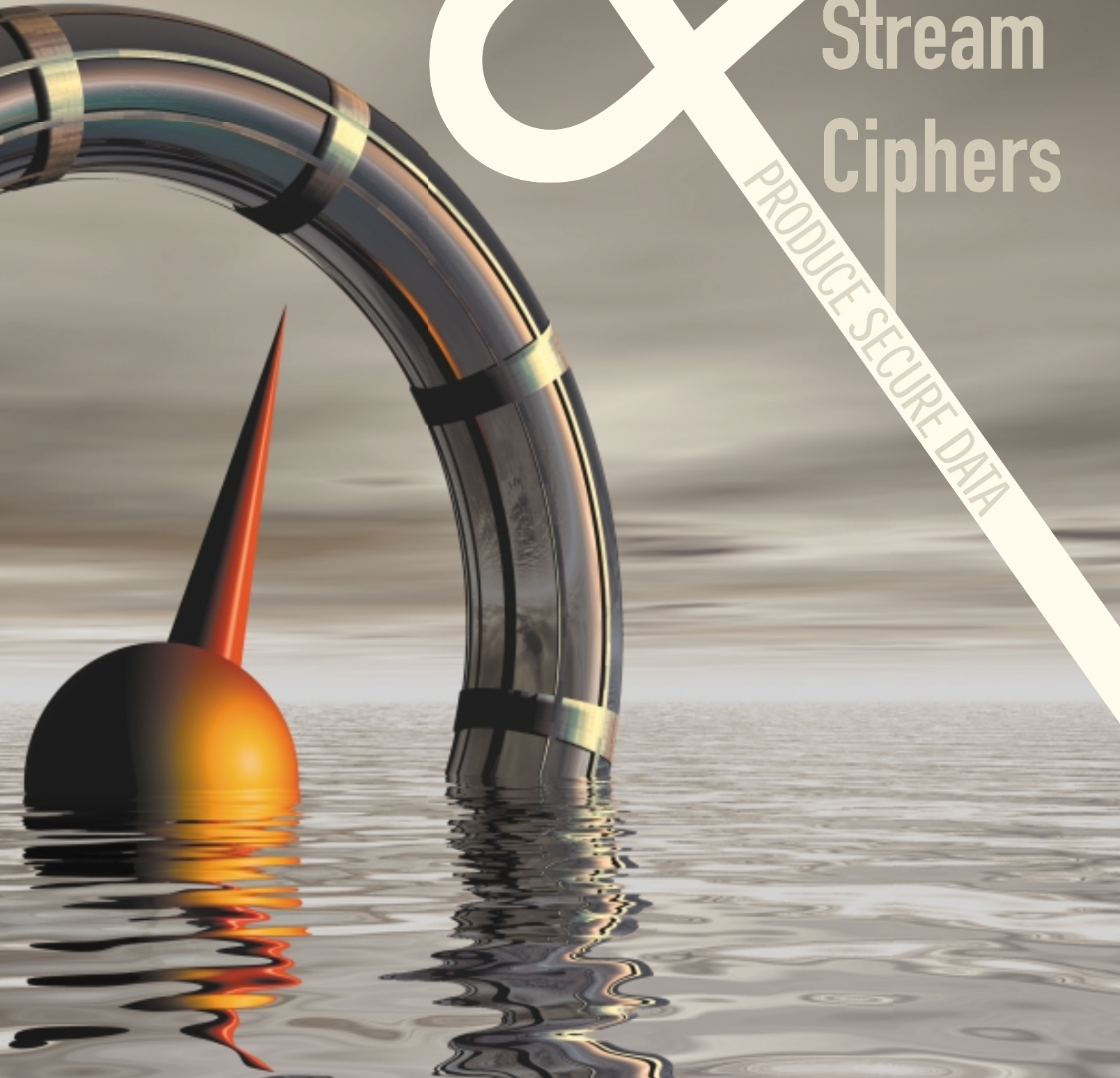


Java



Stream Ciphers

PRODUCE SECURE DATA



When Java sockets came out, the encryption could be easily managed through a stream of data. After getting data in a socket stream and encrypting the stream as it passed through the Internet, I was hooked on Java. While C++ was prevalent, it didn't seem to have streaming algorithms ingrained in the language as well as Java did. Java created a secure programming language that separated itself from the operating systems and network internals, while the streams created a layer to process a continuous wave of data that could be encrypted and further the evolution of programming techniques.

In this article I discuss streams from the cipher perspective and provide an example of how to design and build a stream algorithm so you can practice proper techniques rather than rely on the technology to do it for you (an Ant script to deploy is included with the source code, which can be downloaded from www.sys-con.com/java/sourcecec.cfm). The basic terminology to remember throughout the article is that a cipher is an encryption/decryption algorithm, and a stream is data processed a piece (either bit or byte) at a time. Knowing only those terms, you can build on the rest.

Theory

If you understand a lot about pattern matching and encryption techniques, and are familiar with *Applied Cryptography* by Bruce Schneier, this section may seem too simple. For the rest of you, I'd like to start with the basics of streams and encryption (I've also written a book, *Java Security Solutions*, that provides good information on the topic).

As I mentioned earlier, a stream is data that's processed one bit or, more likely, one byte at a time. It's worth noting

Key Patterns

To make the key pattern simple, if my key was all 0s and XORed with the plaintext, then the ciphertext would be the same as the plaintext. The cipher is useless. Now if the key contained only 1s and XORed with the plaintext, it would be easy to see the difference. If there was more of a mix of 1s and 0s throughout, it would become more difficult to see any relationship between the plaintext and ciphertext.

Anyone who has access to the key can easily decrypt a message. Finding the algorithm that was used to encrypt is not complex, because like a virus, an encrypted message may also contain signatures that can describe its originating algorithm. The size of the key will determine how easy or difficult it is to break an encryption simply because a smaller size key has fewer possible choices, while a larger key has more.

To understand the concept of a stream cipher, part of the basics, let's discuss the theory of a key stream, sometimes called a running key. A key stream, in theory, is a continuous key that's constantly and randomly generated to produce the ciphertext. In other words, each key byte generated is XORed with a plaintext byte to produce a ciphertext byte for the size of the plaintext. In theory, if we have true randomness and the key is infinite, then the encryption could not be broken. The larger the key, the more secure the ciphertext, because there are more permutations of a key that have to be broken. The more random the key, the harder it is to break, because any pattern becomes harder to reverse.

The algebraic notation for the previous discussion can be represented as $c_i = p_i \oplus k_i$. The symbol c_i stands for the ciphertext at index i , the symbol p_i stands for the plaintext



Java created a secure programming language that separated itself from the operating systems and network internals"

that most algorithms will only work on a byte as a whole, not at a bit level. A stream cipher is both a decryption and encryption algorithm for streams. Encryption is used to change readable text – plaintext – into a nonreadable form – ciphertext. Decryption does the reverse. While most ciphers follow various block cipher modes based on the original DES, a true stream cipher (like RC4) comes in handy with unknown block sizes.

For those who are unfamiliar with ciphers and keys, a cipher is the engine that decrypts and encrypts data. The key is the extra data needed for the engine to complete the task. In the early days of cryptography, only a handful of people knew the algorithms and only they could encrypt and decrypt data. As time progressed, most algorithms became published specifications that anyone could access, and the key became the missing piece to ensure that not everyone could encrypt and decrypt the data. The key is a very important element that must be protected at all costs, especially if the key is symmetric. A symmetric or secret key is one in which the same key is used for encryption and decryption.

data at index i , and k_i stands for the key at index i . As we'll see in the RC4 algorithm, XOR is great because the same algorithm can be used in reverse. That is $p_i = c_i \oplus k_i$, which means I can find the plaintext by XORing the ciphertext and the original key.

One of the practices that evolved from running a key theorem is using a product of one ciphertext byte as the key for the next plaintext byte. Another evolution of the running key is the idea of using the key to generate a larger set of keys by hashing initialization data to generate an S-box (a substitution box). S-boxes are discussed later, but can be described as creating a vector from a key to manipulate the data in an algorithm.

When all is said and done, we need to have a key (say 40-bits in some cases) with as few patterns as possible and the key needs to be kept secure. If you remember anything from this article, safeguarding the key must be the highest priority, since it controls access just like the keys to your car or house. The other point to remember about keys is that size does matter.

Develop IT.



Businesses need your help.

Even as IT budgets are shrinking, companies rely on developers like you to help them solve problems—and there are more ways than ever to get the job done. Java, .NET, PHP, Web services, open source, and other emerging technologies let you build and integrate applications and functionality custom-tailored to today's most pressing business needs.

7 themes that are transforming IT:

As new ideas and technologies arise throughout the IT industry, seven key themes have emerged that will change the way companies do business:

- Web Services
- Windows Platform
- Open Source and Linux
- Wireless and Mobility
- The Digital Enterprise
- On-Demand Computing
- Security

Get the big picture. At COMDEX, you'll see all the options side-by-side in a single vendor-neutral environment. Top-tier conference sessions, tutorials, keynotes, and Innovation Centers will show you how to make the latest in IT relevant to today's business needs.

See it all at COMDEX. Hear what's coming next from an elite lineup of speakers from Amazon.com, AT&T Wireless, Computer Associates, Hewlett-Packard, Intel, Microsoft, Nokia, Siebel, Sun Microsystems, Symantec, and more. Share ideas with industry visionaries, business leaders, and thousands of your peers. And experience the vitality of an \$870 billion industry being reborn.

Visit COMDEX.com for details.

Save up to \$200.
Register by October 17.

Use Priority Code: ADBM3NC

Las Vegas
Convention Center

November 17-20, 2003
Educational Programs: November 16-20, 2003

COMDEX
LAS VEGAS 2003

THE GLOBAL TECHNOLOGY MARKETPLACE

Copyright © 2003 MediaLive International, Inc., 795 Folsom Street, San Francisco, CA 94103. All Rights Reserved. MediaLive International, COMDEX, and associated design marks and logos are trademarks or service marks owned or used under license by MediaLive International, Inc., and may be registered in the United States and other countries. Other names mentioned may be trademarks or service marks of their respective owners.

 MediaLive
INTERNATIONAL

Practical


In the previous section, I expressed the need for the key and its randomization. The providers of Java realized this need, and came up with a more random number generator than most libraries provided. Therefore they produced the `java.security.SecureRandom` class to reseed the generator. The idea is to create a variation of different keys that no one could guess at. Many random generators are not truly random, and some keys could be guessed by knowing the generator. Random generators use a seed to help give it something different to generate a new number. Many may use the time of day as a seed. Others may even randomize the seed, but if there is a flaw in the randomization, the seed won't be any better. The `SecureRandom` class reseeding process does not initialize the random number generator, but factors the initial seed with the next to produce a new seed. For this reason alone, some may consider using Java for their encryption needs.

Other utilities that play a big role in secure programming in Java are the `KeyStore`, the `jarsigner`, and the security manager. While this article is too brief to describe these utilities in detail, you need to know that Java provides a utility called the `keytool` to store keys in a secure store, that Java has a `jarsigner` utility to sign a JAR file so it can't be written into without a certificate, and Java has a security manager that can control which resources can be accessed during runtime using a security policy. These utilities can control access to vital resources and data. I'd like to note that these resources come out of the box in Java as well as many encryption algorithms and, again, this is a benefit of using Java.

The de facto stream algorithm is RC4. RC4 stands for Ron's code number 4, Ron being Ron Rivest, the "R" in RSA. Since it's the de facto stream algorithm, I'll use it as an exam-

ple to design, build, and deploy a stream algorithm. The reason you should understand cipher algorithms and their uses is not just to know how to use them, but to understand when to use them, their vulnerabilities, strengths, and how to develop your own algorithms.

After spending many years as a consultant, I've heard programmers proclaim, "I just need to know how to use it, not what it does; there are builders of the JCEs who are concerned about those talks." Yet, I have gotten a lot of consulting work reworking some organizations' code, usually



Not understanding a cipher algorithm in enough detail could make misusing the algorithm worse than not having an algorithm at all"

because someone didn't understand the algorithm correctly. Just as an e-commerce programmer may understand the internals of JSPs and EJBs, a security programmer needs to know the internals of RC4, RSA, and other algorithms. From the IT security officer's point a view, programmers should be able to give the reasoning, strengths, weaknesses, and history behind the algorithms that they're using. Not understanding a cipher algorithm in enough detail could make misusing the algorithm worse than not having an algorithm at all.

RC4

RC4 is the de facto stream algorithm that was made public by an anonymous cypherpunk contributor in 1994. The knowledge of the algorithm was made public, but I believe its commercial use is still licensed through RSA. I'll leave the reader to contact RSA before using it commercially. Examples

How to Get a Service Provider Certificate

Here are simplified steps for getting a service provider certificate from Sun; of course, the Java 1.4.1 SDK documents go into a lot more detail. You'll need time to get this certificate from Sun. The following steps take you through the process:

1. Get a Code-Signing Certificate:
 - Use the `keytool` utility to generate a DSA key pair.
 - Use the `keytool` utility to generate a certificate-signing request.
 - Send the CSR (Code Signing Request) and other information to the JCE Code Signing Certification Authority at Sun.
 - After receiving the certificate from Sun, import the certificate using the `keytool` utility.
2. JAR the provider class (i.e., `RichProvider`) code using the `JAR` utility.
3. Sign the JAR (i.e., `richprovider.jar`) trusted certification with the certificate from Sun using the `jarsigner`.
4. Ensure that the JAR (i.e., `richprovider.jar`) is in the `%JAVA_HOME%\jre\lib\ext` path.
 - You may install the provider in a different classpath location, but it may require setting up a separate `java.policy` file. The default `java.policy` file is already set for the above directory by default.



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...

New Version 5.0.1

- **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!



International Conference & Expo

Edge 2004 EAST

Development Technologies Exchange

February 24-26, 2004

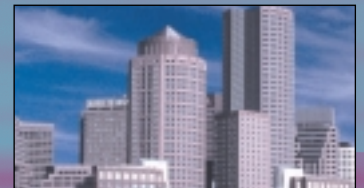
Hynes Convention Center, Boston, MA

ARCHITECTING JAVA, .NET, WEB SERVICES, OPEN SOURCE, AND XML

Addressing:

- ▶ Application Integration
- ▶ Desktop Java
- ▶ Mobility
- ▶ ASP.NET
- ▶ VS.NET
- ▶ JavaServer Faces
- ▶ SOA
- ▶ Interoperability
- ▶ Java Gaming

Register By
November 21, 2003
Up To
SAVE \$400



For more information visit
www.sys-con.com
or call
201 802-3069

Over 200 participating companies will display and demonstrate over 500 developer products and solutions.

Over 3,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 100 of the latest sessions on training, certifications, seminars, case-studies, and panel discussions promise to deliver real world benefits, the industry pulse and proven strategies.

Full Day Tutorials Targeting

- Java • .NET • XML
- Web Services • Open Source

Conference program available online!
www.sys-con.com/edgeeast2004

Contact information: 201 802-3069 • events@sys-con.com



J2ME



J2SE



J2EE



HOME

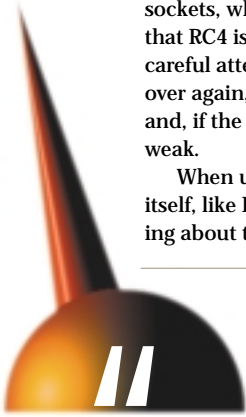
of RC4's commercial uses include Oracle and wireless technologies. I'd imagine that's because of the small and large sets of data needed in these products.

Unlike a stream cipher, a block cipher uses chunks of data, a block, and usually 64 bytes to process through the cipher. If the block ends at less than 64 bytes, the algorithm pads the remaining block. For data that may be a few bytes, this may seem like a lot of overhead. For data that's time-consuming with a lot of I/O, the breaking up of blocks may seem to take up a lot of time. The solution to many may be to use a stream that handles any size data and is quick to process. Some of the places that a stream cipher may be a detriment would be using it for document files in which you wouldn't want plaintext and ciphertext lengths to match. I tend to use stream ciphers with stream I/O, especially Java sockets, when speed is important. Some users of RC4 state that RC4 is 10 times faster than DES. When using RC4, pay careful attention to the keys. If the same key is used over and over again, it could be compromised by constant observation and, if the key is not adequately randomized, it could be weak.

When using a cipher in Java, understanding the cipher itself, like RC4, is only a piece of the puzzle. An understanding about the Java Cryptography Extension (JCE) and service

At(new RichProvider(), 1);. Either way will give a provider interface that's defined by its class and position in the provider chain. Where the provider falls in the chain is also important. There could be three RC4 cipher algorithms defined with the alias RC4, but the executing programs will pick up the first alias in the chain. I defined my provider as the first, so if there are other RC4 algorithms with the same alias, mine will be executed, not the others. If hackers place their provider in the chain while code is executing, it could give them a doorway to your data.

Using the example class `com.richware.cipher.RichProvider`, the provider class is simple and there are a few things to remember. I declared the class as a final class so as not to allow the class to be extended; the class is extended from the `java.security.Provider` class shipped with the Java 1.4.1 SDK. The `RichProvider` registers information about itself to describe its origins like its name, version, and info. If a programmer is executing providers and possibly one that they downloaded, this is important information since it allows you to discover the origin of the provider. A lot of security goes into the provider interface because the valuable data of an organization that they encrypt could be sent through the Internet through a rogue provider.



When using a cipher in Java, understanding the cipher itself, like RC4, is only a piece of the puzzle"

providers becomes paramount when using any cipher. A large part of understanding how a provider is accessed through the provider chain, how to access the provider, and how the algorithm is used in `KeyGenerator` and in `CipherSpi` is crucial. Understanding these concepts is important because programmers may be using a service provider without understanding the origin of the cipher they're using. In other words, programmers need to understand how to prevent Trojans and backdoors by understanding the origins of what their code is using.

All the key generators and ciphers in Java are built using the Service Provider Interface (SPI) layer. The idea of an SPI layer is to provide vendors with the ability to create their own algorithms with the use of a common interface. Since Sun supplies this interface; it allows others to commercially produce extensions that could work within the 1.4.1 SDK while not having to be built with the SDK. This article provides the code necessary to create a provider (code can be downloaded from www.sys-con.com/java/sourcec.cfm). All providers are registered with Sun to ensure that Sun knows who is integrating and interfacing into the 1.4.1 SDK.

Provider

The first step in using a provider is to get the SPI loaded in the provider chain. One way to load a provider is to place the provider in the `%JAVA_HOME%\jre\lib\security\java.security` file as a line item like `security.provider.1=com.richware.cipher.RichProvider`. Another way is to load it at runtime in code as shown in the `com.richware.cipher.TestRC4Cipher` class using the code `Security.insertProvider`

Another piece of the provider is that it associates aliases to classes, usually both a `KeyGenerator` and a `Cipher` alias depending on the algorithm. However, this is totally dependent on matching a corresponding key type to the algorithm. For example, I used the following code:

```
AccessController.doPrivileged(
    new PrivilegedAction() {
        public Object run() {
            put( "Cipher.RC4",
                "com.richware.cipher.RichRC4Cipher" );
            put( "KeyGenerator.RC4",
                "com.richware.cipher.RichRC4KeyGenerator" );
            return null;
        }
    } );
```

This code simply means that when I pass RC4 in a `KeyGenerator`, it calls my `KeyGenerator` service provider code `com.richware.cipher.RichRC4KeyGenerator`. It will likewise call the provider's corresponding code when I pass RC4 in a `Cipher` instance. The code fragment executes this code in a block as a privileged action, which gets the JVM's Security Manager involved. All provider code must be signed in a Java Archive (JAR) file with a certificate from Sun so that security providers can possibly be tracked. In my example, when the `richprovider.jar` gets loaded, it has to be authenticated with a trusted certificate. You have to use the `keytool` to get the trusted certificate and the `jarsigner` utility to sign it in the provider's JAR file. Take a look at the sidebar "How to Get a

Service Provider Certificate” for a set of simplified steps.

Looking at the steps in the sidebar, it’s obvious that there are a lot of security provisions and traceability for using Java JCE providers. It was a lot different in the C++ days. In those days we just added a Dynamic Link Library (DLL) to the System32 path in Windows or a library in Unix. However, not to be preachy about the robustness of Java, you can examine the origin and execution of the JAR file. For instance, when in doubt about a JAR file, just move the JAR and isolate the execution of it to another system to examine. A trace of the JVM tracing into the JAR could be done to see if the JAR is Trojaned, but that is another discussion.

When using some of the more native libraries, it becomes more difficult to trace for Trojans through the libraries because it requires an understanding of the operating system that the native calls are integrated within. Some security consulting involves isolating the signatures for Trojans on libraries stored in the computer. These techniques help in host-based intrusion detection.

KeyGeneratorSpi

The purpose of the provider is to securely associate the correct KeyGenerator and Cipher code to the algorithm being called by the application. Most KeyGenerator code is responsible for generating a random key. Most of the differences rely on the key size. The RC4 algorithm allows for a key size of 1–256 bytes to match up to the size of the S-box. Now there are export restrictions, so when exporting to other countries, it may be necessary to limit the key size. You must contact the Department of Commerce for current export restrictions.

The code checks the key size in bits and, if it’s the wrong size, it will throw an exception, otherwise it will generate a

key based on the size. Most of the work is ensuring that the correct key size, given in bits and generated in bytes, is returned as a SecretKeySpec class.

Notice in Listing 1 that the SecureRandom class is used to create the random number key. The SecretKeySpec class is returned because the key is a secret key. One of the features of the code is that if you’re not happy with Java’s SecureRandom class and feel that you can build a better one, you can extend it and pass it in the KeyGenerator class to use it instead. A fragment of the RichRC4KeyGenerator code is provided in Listing 1.

CipherSpi

The CipherSpi is also started from the provider when it associates the Cipher.RC4 with the com.richware.cipher.RichRC4Cipher class. The key is generated in the previous section; now it’s time to encrypt or decrypt the message. The CipherSpi normally takes in the mode of operation for the cipher engine, like the operation to encrypt versus decrypt. In the RC4 algorithm, there’s no difference between encryption and decryption except for the input data. When the RC4 engine is initialized, it will first build its S-box. Many algorithms have multiple S-boxes, but RC4 has one array of S-boxes from 0 to 255. Listing 2 provides an example.

The building of an S-box involves manipulating the key from an initialized S-box to produce a new substitution box to be used in the RC4 algorithm. Simply put, an S-box is built from a key and some initialization code like a new key that cannot be deciphered. The idea is to build an S-box to swap data from a known index into an unknown index to avoid Gaussian elimination in trying to reverse the algorithm. This is accomplished by using the random key to define the posi-

WebCharts3D

One demo worth your download

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.**



Rich Helton has worked on computer systems since 1982 and entered the private sector in 1990 as a lead computer architect with OmniPoint Data Corporation, the inventors of PCS. Since then, he started consulting as a security architect, providing many organizations with secure Enterprise and Network solutions. He is the co-author of *Java Security Solutions* and the *BEA WebLogic Server Bible*.

richware@earthlink.net

tion of the next swap with the `index2` variable. The counter variable, along with the `index1` variable, ensures that all the S-box bytes are swapped at least once during calculations. The idea is simply to try to avoid any pattern and factoring of the S-box while having the same key produce the same S-box.

After the S-box is built from the key, the RC4 algorithm can be used to encrypt or decrypt the data. Listing 3 demonstrates the RC4 cipher.

From the code, you can see that each output byte of the RC4 algorithm is the product of each input byte that is XORed with an S-box value with the `xorIndex`. First, an `x` and `y` index is selected. The `y` is a product of the S-box from the `x` index. The two S-boxes are swapped. Then an S-box is selected, a `xorIndex` that is the product of two other S-boxes that are symbolized by `x` and `y` indexes. Again, the idea is to keep swapping the S-boxes and index to make the value and location difficult to produce by factoring. Finding ways to make finding patterns and reverse factoring difficult becomes the guideline for developing ciphers.

Testing the Program

The Test program is straightforward. The `com.richware.cipher.TestRC4Cipher` class will load up the `RichProvider` class and generate a key of a `SecretKeySpec` class that is 128 bits with the following:

```
kg.init(128);
Key secretKey = kg.generateKey();
```

The program encrypts the message "This is a test, hackers beware," that is 31 bytes to a 31 byte encrypted message sim-

ilar to "cUi8DZfy+IQti6xl4Z4FhzRZl2mY2Pa7RmZygn VXnA==" depending on the key. After encrypting, I decrypt and compare the output to the original message to see if anything changes. Listing 4 provides the code fragment that accomplishes this.

Included in the source code is a "-v" option that puts the test code and provider in a verbose mode for readers who might want to trace the provider calls and S-box information. A point to note in Listing 4 is that the same secret key used for encryption is used for decryption. When building the `richprovider.jar`, it won't work without being signed by the Sun certification. A "java.security.NoSuchProviderException: JCE cannot authenticate the provider RichWare" exception should appear without the certification.

Summary

This article introduces a stream algorithm to help you understand the proper techniques to produce better and more robust algorithms in the future. Many standard cipher algorithms have been around for decades with little to add to them. You would think that it was due to the fact that they have not been broken, but some algorithms like DES have been broken many times. Understanding the strengths, weaknesses, and how the algorithms work may help us produce more secure data for our corporations and ourselves. ☛

References

- Schneier, B. (1996). *Applied Cryptography, Protocols, Algorithms, and Source Code in C, Second Edition*. John Wiley & Sons.
- Helton, R., and Helton, J. (2002). *Java Security Solutions*. John Wiley & Sons.

INT, inc.
www.int.com

Listing 1

```
protected void engineInit( int i, SecureRandom
_securerandom ) {
    if ( ( i % 8 != 0 ) || ( i < 8 ) || ( i > 2048 ) )
        throw new
            InvalidParameterException(
                "Keysize must be multiple of 8, and can only range
from 32 to 448 (inclusive)" );
    else {
        keysize_ = i / 8;
        engineInit( _securerandom );
        return;
    }
}
/**
 * Method engineGenerateKey
 * @return the RC4 Secret Key
 */
protected SecretKey engineGenerateKey() {
    if ( random_ == null )
        random_ = new SecureRandom();
    // Create a temporary storage for the key
    byte temp[] = new byte[keysize_];
    // Randomly generate the key
    random_.nextBytes( temp );
    // Return a Key Spec with the Key and Key Type
    return new SecretKeySpec( temp, "RC4" );
}
}
```

Listing 2

```
protected void prepare_key( Key _key )
throws InvalidKeyException {
    /* Fill the S-box with the key
    * Key Setup */
    byte[] userkey = _key.getEncoded();
    if ( userkey == null )
        throw new InvalidKeyException( "Null user key" );
    // Check key length
    int len = userkey.length;
    if ( len == 0 )
        throw new InvalidKeyException(
            "Invalid user key length" );
    // Reset x and y
    x = y = 0;
    // Populate the the initial Sbox with 0 through 255
    for ( int index = 0; index < 256; index++ )
        sBox[index] = index;
    // Initialize variables
    int index1 = 0;
    int index2 = 0;
    int temp;
    /* A temporary value will be taken from the S-Box
    * that will be executed from 0 to 255.
    * The key value is indexed from the length of the
    key.
    * The S-box indexed from the counter will be
    swapped with the S-box
    * indexed from the temporary value.
    * The swapping will happen until all the S-boxes are
    walked from 0 to 255.
    * If the same key is passed through the Sbox, it
    will generate the same Sbox.
    */
    for ( int counter = 0; counter < 256; counter++ ) {
        index2 =
            ( ( userkey[index1] & 0xFF ) + sBox[counter] +
index2 )
            & 0xFF;
        // Swap the byte
```

```
temp = sBox[counter];
sBox[counter] = sBox[index2];
sBox[index2] = temp;
index1 = ( index1 + 1 ) % len;
}
}
```

Listing 3

```
protected void rc4( byte[] in, int inOffset, int inLen,
byte[] out, int outOffset ) {
    /* The byte is XORed with the plaintext to produce
the ciphertext
    * The byte is XORed with the ciphertext to produce
the plaintext
    * The algorithm is symmetric, meaning this function
will work for both
    * encryption and decryption
    */
    int xorIndex;
    /* The byte is XORed with the plaintext to produce
the ciphertext
    * The byte is XORed with the ciphertext to produce
the plaintext
    * The algorithm is symmetric, meaning this function
will work for both
    * encryption and decryption
    */
    int temp;
    for ( int i = 0; i < inLen; i++ ) {
        x = ( x + 1 ) & 0xFF;
        y = ( sBox[x] + y ) & 0xFF;
        temp = sBox[x];
        sBox[x] = sBox[y];
        sBox[y] = temp;
        xorIndex = ( sBox[x] + sBox[y] ) & 0xFF;
        out[outOffset++] = ( byte ) ( in[inOffset++]
^ sBox[xorIndex] );
    }
}
```

Listing 4

```
Cipher cipher =
    Cipher.getInstance( "RC4", "RichWare" );
Cipher cipher2 =
    Cipher.getInstance( "RC4", "RichWare" );
// Init the Cipher with Key, mode and random genera-
tor
cipher.init( Cipher.ENCRYPT_MODE, secretKey, _random
);
// Encrypt the message with the secret key.
byte[] encryptedMessage =
    cipher.doFinal( messageBytes, 0,
messageBytes.length );
// Init the Cipher2 with Key, mode and random genera-
tor
cipher2.init( Cipher.DECRYPT_MODE, secretKey, _random
);
// Decrypt the message with the secret key.
byte[] decryptedMessage =
    cipher2.doFinal( encryptedMessage, 0,
encryptedMessage.length );
String decryptedMessageString =
    new String( decryptedMessage, "UTF8" );
/* Check that the decrypted message and the origi-
nal
* message are the same.
*/
if ( decryptedMessageString.equals( message ) )
    System.out.println( "\nTest succeeded." );
else
    System.out.println( "\nTest failed." );
```



Glen Cordrey
J2ME Editor



Quality Is Job *n*?

At a training session I recently attended, a presenter mentioned that his cell phone crashes whenever he runs a simple MIDlet that he wrote. While it may have been inevitable that poor-quality environments would make it onto J2ME platforms, it's still distressing to see some J2ME development proceeding down the trail blazed by the megacorp in Redmond.

Now I am not one of those who despise Microsoft. Microsoft is not inherently evil, and both good and bad have come out of it, as is true of just about any corporate entity. But I do feel that Microsoft's sins have been egregious in the area of software quality, and I fear that it has inured us to accept software of far lesser quality than we should accept.

When your desktop PC offers up the blue screen of death, you may be annoyed and curse the denizens of Redmond, but most people have come to accept occasional freezes and splats as one of the costs of computing (or, at least, the cost of computing with Windows). But there's no reason we should carry those diminished expectations to mobile devices.

Because the primary purpose of PCs is computing, it may be easy to rationalize software failures as the price you pay, but the primary purpose of cell phones is communication. For this reason – or from a different perspective, because anything that impedes making and receiving calls reduces revenue – most cell phone manufacturers and network providers have historically been very restrictive about what, if any, third-party software they allow on their handsets. If customers can't make or receive calls, they'll be more inclined to switch carriers, and they'll rarely make a distinction as to whether it was the hardware, the software, or the network that was at fault.

This mentality contrasts sharply with that of the other major platform for J2ME applications, PDAs, which have from the beginning been first and foremost about applications. Since the early Palm Pilots (and their precursors), the modus vivendi with PDAs has been to

download applications; so where cell phones restricted customer deviation from a defined configuration, PDAs embraced it. Consequently, PDA manufacturers and users may accept the occasional crash, much like they do on PCs, as a risk worth taking. To what degree are we willing to trade off the reliability of cell phones for the convenience that downloadable applications add?

Shouldn't we first ask whether we need to make significant trade offs? Is it not feasible to architect the KVM to be, if not bulletproof, at least robust enough to prevent all but the most onerous and insidious programming errors from significantly compromising the reliability of the platform? Or is it that we could, but it's not considered cost-effective to invest the extra effort to do so? Or is it time-to-market that trumps the delivery of quality software?

Consider some of the vulnerabilities that poor quality software could expose us to as cell phones become more powerful. You'll have financial and other personal information on your smart phone, but will it have enough computing capability to support virus checker, or firewall software powerful enough to thwart sophisticated attacks? Your cell phone's mobility combined with Bluetooth and WiFi capabilities will allow it to be more promiscuous than a desktop or a less-mobile laptop, joining and parting ways with numerous hotspots as you travel around. Does ubiquitous computing mean ubiquitous targets for compromised security?

Many efforts are afoot to address these concerns, and the walled-garden security model of the KVM is a good foundation. But as with building a house, a great blueprint counts for little if the construction is shoddy. What can we do? As developers we can promote high quality in the software we write, and provide informed voices in discussions of the issues and in the development of specifications. As consumers we can vote with our pocketbook, eschewing OEMs and network providers who offer substandard services. But ultimately it's the marketplace that will determine the quality provided. ☺



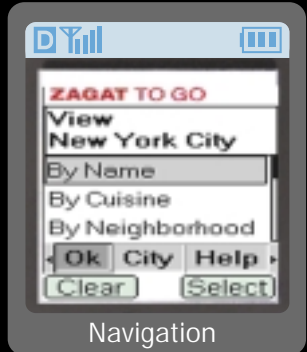
Quality Is Job *n*?

At a training session I recently attended, a presenter mentioned that his cell phone crashes whenever he runs a simple MIDlet that he wrote. While it may have been inevitable that poor-quality environments would make it onto J2ME platforms, it's still distressing to see some J2ME development proceeding down the trail blazed by the megacorp in Redmond.

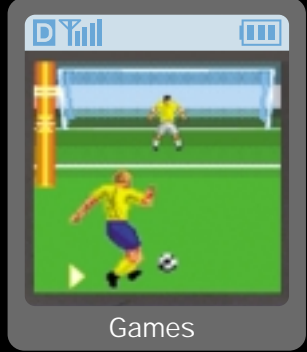


Glen Cordrey is a software architect working in the Washington, DC, area. He's been using Java for five years, developing both J2EE and J2ME applications for commercial customers.

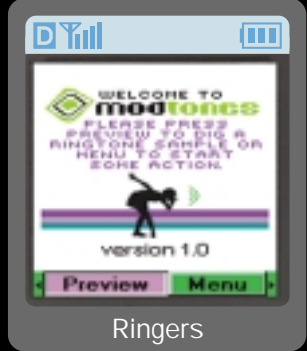
glencordrey@sys-con.com



Navigation



Games



Ringers

MILLIONS OF
POTENTIAL CUSTOMERS.
ANY TAKERS?



It's called "Get It Now.SM" And it's the name Verizon Wireless, the nation's largest wireless carrier, has chosen for its new nationwide, BREW-enabled service. Dozens of BREWTM applications are now heading directly from the hands of developers into the hands of consumers with the help of the BREW Distribution System – and the financial rewards for these developers are returning just as fast. And as more carriers prepare to launch their BREW-enabled services, BREW applications such as games, email, news, weather, stock trades, position location and ringers are finding a potential market of literally millions upon millions of users. If you aren't developing for BREW, you aren't developing to your potential. To get started, visit www.qualcomm.com/brew/jdj.





THE LOCATION API

by Sven Haiges

Simplify access to
mobile positioning
methods

T

he Location API (JSR 179) was accepted by the Executive Committee for Micro Edition of the Java Community Process in June 2003. It provides an abstract interface for access to location-based information, such as the current coordinates of the mobile terminal independent from the underlying positioning method used.

Mobile Positioning 101

Mobile positioning in general is the process of determining the location of a mobile terminal; the result of this process is the mobile location, which is the current location in terms of coordinates of the mobile terminal. The mobile location can be more or less accurate, depending on the mobile positioning method used.

The large number of existing methods that can be used for mobile positioning can be divided into two broad categories: network-based and handset-based. The first category, network-based positioning, is made up of many methods that reach from relatively simple COO (cell of origin) methods to more complex methods like TDOA (Time Difference of Arrival). The latter category, the handset-based positioning methods, allows locating the terminal without the help of the mobile communication network (for example, the GSM or CDMA network). GPS – the Global Positioning System – is the most prominent member of that category.

Besides those two categories, there are also hybrid methods that unite network-based and handset-based methods. AGPS (Assisted GPS) is one, where you try to speed up the initialization time of the GPS receiver by using information coming from the mobile network.

The following sections explain the most common positioning methods in more detail. Figure 1 provides an overview of the described methods.

Handset-Based Methods

- **Global Positioning System (GPS):** This system was developed by the U.S. Department of Defense and is still maintained by them. Originally developed only for the military, today this positioning system is widely used by civilians. The system is based on signals that are transmitted from 24 satellites that GPS receivers on earth use to calculate the current location. Each GPS receiver needs to be in con-

tact with four satellites at the same time to be able to determine the user's longitude, latitude, and altitude.

Among the advantages are the relatively exact positioning result (close to 10–30 meters) as well as the possibility of determining the altitude at which the terminal is located. A serious disadvantage is the lack of reception of GPS signals within buildings, which can make the positioning process impossible.

- **Enhanced Observed Time Difference (E-OTD):** This method is often referred to as the opposite of the TOA (Time of Arrival) method. Instead of measuring the time differences of signals that are transmitted from the terminal to the base station, the terminal calculates the location. Therefore, special software has to be installed in the mobile terminal. In addition, so-called "Location Measurement Units" have to be installed at each base station. The accuracy of E-OTD is relatively high, although the best results that are close to GPS can only be expected in urban areas, where many GSM cells are available.
- **Subscriber Identity Module Toolkit (SIM):** Another handset-based positioning method is via the SIM Toolkit. The STK is an API that allows communication with the SIM smart card, which many mobile phones have from applications that were installed on the handset. The quality can be as bad as the COO method, but can also be improved by some algorithms that are stored on the SIM and by some extra features that are provided by the network.

Network-Based Methods

- **Cell of Origin (COO):** COO is the easiest and most common method, but also the most inaccurate one. The network determines only the cell from which the user is placing a call or initiating a data transfer and can determine the location based on the known locations for the base stations of the network. Interestingly, this method is already sufficient for most location-based services, and additional calculations like Timing Advance (TA) can further enhance

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 **\$275⁰⁰**
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 **\$350⁰⁰**
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 **\$400⁰⁰**
Pay only \$495 for a 1 year subscription plus **3 FREE CDs**

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

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.

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

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.

Linux Business & Technology

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

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

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

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

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

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

.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

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

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

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm





the quality of positioning.

- **Angle of Arrival (AOA):** This method uses special equipment that has to be installed at the base stations to determine the angle of arrival for the radio signals. With some basic geometric calculations, you can then determine the location of the user with only two base stations receiving its signal.
- **Time Difference of Arrival (TDOA):** This method uses the differences of arrival for the radio signals at the base stations (from the mobile terminal). A minimum of three receiving base stations is required to calculate the location of the user.
- **Location Pattern Matching (LPM):** This complex method analyzes the radio signals and compares them to patterns saved in a database. These patterns include signal reflections and echoes. When a pattern is recognized, the location of the user can be identified. This method can be used only in urban areas, where these signals often occur. In these areas the quality might be better than using other methods; unfortunately this method is not suitable as a general method for rural areas.

Hybrid Methods

- **Assisted GPS (AGPS):** AGPS uses information from the network to more quickly determine the position of the four satellites that it needs to listen to. The network cell distributes the locations of those satellites and can drastically reduce the initialization time that a normal GPS receiver needs. In addition, this method saves battery power as the GPS receiver is only activated on a usage basis.

Other methods of mobile positioning include short-range beacon methods such as Bluetooth or IrDA. Those methods can be used to locate users indoors and the quality of service is good. On the other hand, the setup of those networks is a highly complex task as a new infrastructure is needed and mobile terminals have to be updated as well.

The Location API

Access to the current location of the terminal is of special value for mobile phones and embedded systems. The Location API allows us to access this kind of information through a standard interface. The underlying mobile positioning method does not change the usage of the API but, of course, the quality and amount of information that can be gathered can vary tremendously.

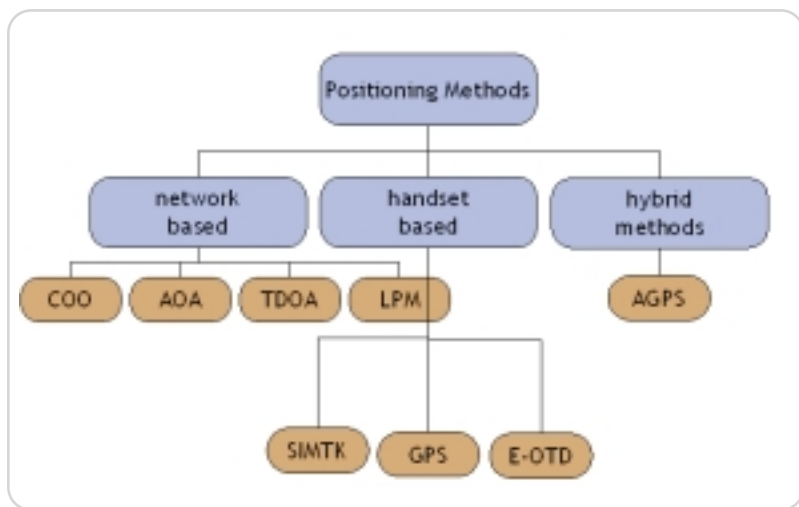


Figure 1 Positioning methods

"The Location API simplifies access to the results of mobile positioning methods"

This JSR is led by Kimmo Löytänä (of Nokia Corporation) and has been accepted in unison by the Executive Committee for Micro Edition, showing strong industry-wide support for this API, in June 2003. The Location API (package `javax.microedition.location`) needs the CLDC 1.1 as an underlying configuration, as support for floating point numbers is necessary for values such as the longitude and latitude. Furthermore, the utilization of the security concept of MIDP 2.0 is recommended in the JSR. If an application wants to make use of the Location API, the user will be prompted for the permissions. He or she can decide to give those permissions to use the Location API on a per-usage basis for the duration of the current session or in general. This inhibits the abuse of this API, as every location request might involve costs to the user.

Figure 2 provides an overview of the main classes involved in the Location API.

The `LocationProvider` is a Singleton class that is the starting point for a request to the location service. The method `getInstance(Criteria c)` receives a criteria object as a parameter to determine the needs of the location request (for example, desired accuracy, whether to include address information, etc.).

Criteria can generally be divided into hard and soft criteria. Hard criteria are:

- Maximum cost per request
- Should the speed be determined?
- Do we need address information or are the coordinates sufficient?

Soft criteria are:

- The horizontal/vertical accuracy of the positioning process
- The degree of power consumption tolerated
- The maximum response time

In addition to blocking a request by using the method `getLocation()` of the class `LocationProvider`, there's also a nonblocking `LocationListener` that can be registered at the `LocationProvider`. The `LocationListener` will then receive the current location information at regular intervals.

Orientation data can also be gathered directly from the `LocationProvider` class and includes the pitch and roll as well as the compass orientation of the terminal.

The response from the `LocationProvider` is an object of the class `Location`, which encapsulates the location information. It includes a `Coordinates` and `AddressInfo` object. `Coordinates` provide information, such as the longitude/latitude values of the location. `AddressInfo` provides additional information like the name of the street, city, etc., if this information has been specified by the criteria and is available.

Landmarks (not illustrated) allow you to classify locations and save them to a landmark database on the mobile handset. The classifications could, for example, include movie theaters, ATMs, or museums. In addition, personal landmarks such as the home location can be saved to this database. The database uses the RMS (Record Management

A new tool for MX professional developers and designers...

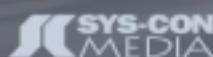


ADVERTISE

Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE

[www.sys-con.com/
mx/subscription.cfm](http://www.sys-con.com/mx/subscription.cfm)
1 (888) 303-5282



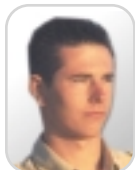
MX
developer's journal





System) that is part of the underlying profile, the MIDP.

That's the theory. Please keep in mind that the actual location information might not include the street name or the orientation of the mobile terminal. The accuracy will vary as well depending on the method used. Although technically feasible with certain handsets (for example, AGPS-enabled devices from Motorola), the quality of information will not always be that good. This API has been designed with the future in mind. As the positioning methods evolve, the accuracy and the amount of information supplied by this API will improve and increase.



Sven Haiges

(www.svenhaiges.de)

is currently completing degrees in computer science and business administration at the University of Furtwangen, Germany, and technology management (MBA) at Simon Fraser University in Vancouver, Canada. As a specialist in the Java programming language, he combines knowledge of both server-side and client-side Java, especially for mobile Java clients.

Sven recently published his first book about the Struts open source Web application framework in Germany.

sh@flavor.de

The example in Listing 1 is taken from the specification and demonstrates the usage of the API.

Here comes the million-dollar question: When will this API be available in mass-market Java phones? Good question, I don't think we'll see the API in a consumer device by the end of this year. The Roadmap 1 (JSR 185 - Java Technology for the Wireless Industry) does not include the API in its recommendation, but the Location API seems to be a hot candidate for the next version of this roadmap. If it will be added to the roadmap, it will most likely be an optional API like the current Multimedia API.

Location-Based Services (LBS)

The Location API will allow the implementation of various location-based services, which can generally be split up into these categories:

- **Public security:** For determining the user's location in case of an emergency, etc.
- **Tracking:** For tracking fleet vehicles, people, etc.
- **Traffic monitoring:** For the fast recognition of jammed highway zones, etc.
- **Location-based information:** Navigation services, city-sightseeing services of mobile yellow pages

A location-based service from Jentro AG, Munich, Germany, will serve as an example. Jentro AG's navigation solution (see Figure 3) was introduced during one of the general sessions at the recent JavaOne conference in San Francisco. Besides a Java-capable mobile phone, the entry version needs only a GPS adapter to turn the mobile phone into a navigation solution. The MIDlet communicates using



Figure 3 Jentro AG Navigation Solution (GPS adapter is at the bottom of the phone)

the serial port with the GPS adapter and sends this information via the MIDP's Generic Connection Framework (GCF) to the server. The server then calculates the routing information between the start and end point and transmits this information to the mobile client. Although Jentro currently does not use the Location API (nor the Multimedia API), as those APIs become available in mainstream phones Jentro will

consider the switch toward those APIs as it speeds up the development of new services.

Conclusion

The Location API simplifies access to the results of mobile positioning methods and allows mobile Java applications to use the location information via standard APIs.

Applications that use this API are more likely to occur in a professional environment first, for example, for fleet management. A developer can target a specific device and a specific network that supports the level of accuracy that the application needs.

As for the mass market, the differences in mobile positioning quality that is likely to occur because of the different underlying mobile positioning methods used will make it more difficult to design applications that use mobile positioning information.

The JTWI 2.0 will hopefully include the Location API as an optional API and provide a clear roadmap for this API. As a developer you don't have to wait until the Location API is available to use mobile location information. This information can be accessed using a GPS adapter (via serial interface) and proprietary APIs. As soon as the Location API is available, switch to using the Location API, as it will speed up new location-based services development and does not lock you into proprietary APIs or solutions.

References

- www.mobilein.com/mobile_positioning.htm
- www.cursor-system.com/products_classic.asp
- www.911dispatch.com/911_file/aoa.html
- www.911dispatch.com/911_file/tdoa.html
- www.911dispatch.com/911_file/lpm.html
- <http://jcp.org/aboutJava/communityprocess/first/jsr179/index.html>
- www.umtsworld.com/technology/lcs.html

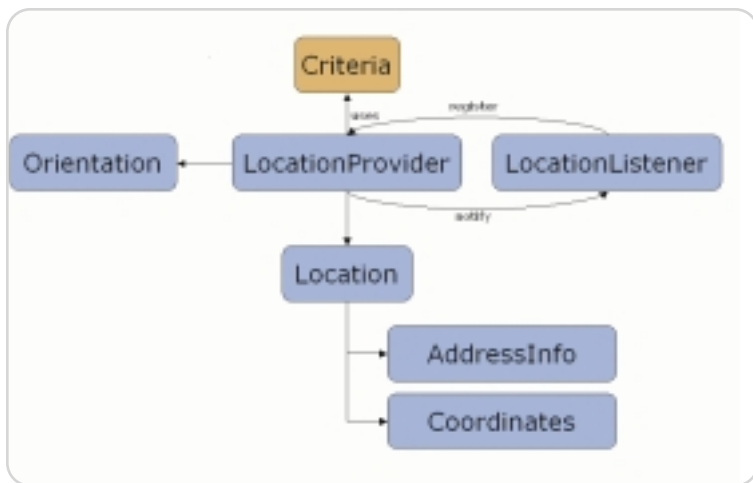


Figure 2 Location API main classes

Listing 1

```

try {
    // Creation of a Criteria-Object
    Criteria cr = new Criteria();
    // Adjusting the horizontal accuracy to 500 meters
    // besides that, no standard values are changed
    cr.setHorizontalAccuracy(500);
    LocationProvider lp = LocationProvider.getInstance(cr);
    // Request the location, we are willing to wait for 60
    seconds
    Location l = lp.getLocation(60);
    Coordinates c = l.getQualifiedCoordinates();
    if (c != null) {
        // use the information...
        ...
    }
} catch (LocationException e) {
    // Display an error message
    ...
}
  
```




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

\$49.99

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201.802.3020 OR

VISIT WWW.SYS-CON.COM

Multiple Inheritance in Java

Of diamonds and dynamic proxies



Dan Haywood

When Sun was designing Java, it omitted multiple inheritance – or more precisely multiple implementation inheritance – on purpose. Yet multiple inheritance can be useful, particularly when the potential ancestors of a class have orthogonal concerns. This article presents a utility class that not only allows multiple inheritance to be simulated, but also has other far-reaching applications.

Have you ever found yourself wanting to write something similar to:

```
public class Employee extends Person,
    Employment {
    // detail omitted
}
```

Here, `Person` is a concrete class that represents a person, while `Employment` is another concrete class that represents the details of a person who is employed. If you could only put them together, you would have everything necessary to define and implement an `Employee` class. Except in Java – you can't. Inheriting implementation from more than one superclass – multiple implementation inheritance – is not a feature of the language. Java allows a class to have a single superclass and no more.

On the other hand, a class can implement multiple interfaces. In other words, Java supports multiple interface inheritance. Suppose the `PersonLike` interface is:

```
public interface PersonLike {
    String getName();
    int getAge();
}
```



Figure 1 The `PersonLike` and `EmployeeLike` interfaces

and the `EmployeeLike` interface is:

```
public interface EmployeeLike {
    float getSalary();
    java.util.Date getHireDate();
}
```

This is shown in Figure 1.

If `Person` implements the `PersonLike` interface, and `Employment` implements an `EmployeeLike` interface, it's perfectly acceptable to write:

```
public class Employee implements PersonLike,
    EmployeeLike {
    // detail omitted
}
```

Here there is no explicit superclass. Since we are allowed to specify at most one superclass, we could also write:

```
public class Employee extends Person implements
    PersonLike, EmployeeLike {
    // detail omitted
}
```

We would need to write the implementation of `EmployeeLike`, but the implementation of `PersonLike` is taken care of through the `Person` superclass. Alternatively we might write:

```
public class Employee extends Employment
    implements PersonLike, EmployeeLike {
    // detail omitted
}
```

This is the opposite situation: the `EmployeeLike` interface is taken care of through the `Employment` superclass, but we do need to write an implementation for `PersonLike`.

Java does not support multiple implementation inheritance, but does support multiple interface inheritance. When you read or overhear someone remark that Java does not support multiple inheritance, what is actually meant is that it does not support multiple implementation inheritance.

Stay Adaptable

Suppose then that you have the concrete implementations `Person`, which implements the `PersonLike` interface, and `Employment`, which implements the `EmployeeLike` interface. Although only one can be selected to be the superclass, it would be useful to somehow exploit the other implementation.

The easiest way to do this in Java is by applying the (Object) Adapter pattern. If we make `Person` the superclass, we can use `Employment` using an object adapter held within the employee:

```
public class Employee extends Person implements
    PersonLike, EmployeeLike {
    private EmployeeLike employment = new
        Employment();
    public float getSalary() { return
        employment.getSalary(); }
    public java.util.Date getHireDate() {
        return employment.getHireDate(); }
}
```

For each method of `EmployeeLike`, the employee delegates to the object adapter. This helps motivate the decision as to whether `Person` or `Employment` should be the superclass; choose the one with the most methods as the superclass so there will be less manual delegation code to write when dealing with the other interface.

The Adapter pattern is a fine way to support multiple interface inheritance while exploiting two concrete implementations. Indeed, it's more often the case that an anonymous inner class is used as the object adapter, allowing customization of behavior with respect to the context (of being embedded within a subclass).

However, writing that delegation code is tedious, especially if both interfaces to be implemented have many methods in them. In many cases, we can get Java to do the delegation to the would-be superclass(es) automatically.

Enter Dynamic Proxies

Dynamic proxies were introduced

into Java in J2SE v1.3. Part of the `java.lang.reflect` package, they allow Java to synthesize a class at runtime. The methods supported by this synthesized class are specified by the interface (or interfaces) that it implements. The implementation is taken care of through an invocation handler (`java.lang.reflect.InvocationHandler`) that is handed an object representing the method being invoked (`java.lang.reflect.Method`). As you can see, dynamic proxies use heavy doses of the Java Reflection API.

This then is the key to simulating multiple implementation inheritance within Java. We can write a custom `InvocationHandler` that is constructed with a set of classes; these represent the superclasses of the subclass to be synthesized. The interface(s) of our subclass will be the union of the interfaces implemented by these superclasses. Our `InvocationHandler` will instantiate instances of these superclasses so that it can delegate to them. We then arrange it so that the invocation handler, on being given a method to be invoked, will reflectively invoke the method on the appropriate superclass object instance. (There must be one; remember the subclass's interface is derived from the superclass's, so at least one superclass must be able to handle the method invocation.)

To make things simple, we can make our `InvocationHandler` implementation also return the proxy. In other words, the invocation handler can act as a factory, able to return instance(s) of the synthesized subclass that will delegate to the superclass instances. We call our invocation handler implementation `DelegatorFactory` for this reason:

```
// imports omitted
public final class DelegatorFactory
    implements InvocationHandler {
    public Object getObject() {
        return Proxy.newProxyInstance(
            this.getClass().getClassLoader(),
            getSupportedInterfaces(),
            this);
    }
}
// code omitted
}
```

The supported interfaces of the resultant object are derived from the superclasses provided in the `DelegatorFactory`'s constructor:

```
// imports omitted
```

```
public final class DelegatorFactory implements
    InvocationHandler {
    public DelegatorFactory(final Class[]
        ancestors) {
        // implementation omitted
    }
}
// code omitted
}
```

There is more to `DelegatorFactory` as we shall soon see, but we now have enough to simulate multiple implementation inheritance. Going back to the question first posed, instead of:

```
public class Employee extends Person,
    Employment {
    // detail omitted
}
```

followed (presumably) by:

```
Employee employee = new Employee();
```

We can instead write:

```
Object employee =
    new DelegatorFactory(
        new Class[] {
            Person.class,
            Employee.class
        }
    ).getObject();
```

Although the syntax is somewhat different, the same essential information is being provided. That is, the concrete implementations are provided in `Person` and in `Employment`. This object will use the implementation of `Person` if invoked as a `PersonLike`, and the implementation of `Employment` if invoked as an `EmployeeLike`:

```
((PersonLike) employee).getAge();
((EmployeeLike) employee).getHireDate();
```

How Convenient

In the above example, the casts are necessary because the `getObject()` method of `DelegatorFactory` can only return a reference of type `java.lang.Object`. But the clunkiness arises because our original aim of defining the `Employee` class with two concrete superclasses actually does something else as well:

```
public class Employee extends Person,
    Employment {
    // detail omitted
}
```

Not only does this indicate that the implementation of `Employee` should be based on that of its superclasses, it also

defines `Employee` as a type. In other words, it's then possible to write:

```
Employee employee;
```

What is missing in our dynamic proxy solution is this definition of type. Let's first do that in the usual way. As shown in Figure 2, we don't need to use a class though; an interface is sufficient.

As code, this is simply:

```
public interface Employee extends
    PersonLike, EmployeeLike { }
```

There is no detail omitted here; this is our complete definition. Note that `Employee` is now an interface and not a class.

The following will not work, however:

```
Employee employee =
    (Employee)
    new DelegatorFactory(
        new Class[] {
            Person.class,
            Employment.class
        }
    ).getObject();
```

This is because the only interfaces implemented by the dynamic proxy returned by `getObject()` are `PersonLike` and `EmployableLike`. No matter that logically the `Employee` interface does not require any additional implementation from our dynamically created object; `Employee` is not an interface that we can cast to.

However, `DelegatorFactory` does provide an alternative constructor:

```
Employee employee =
    (Employee)
    new DelegatorFactory(
        new Class[] {
            Person.class,
            Employment.class
        },
        Employee.class
    ).getObject();
```

Note the new second argument (`Employee.class`) to the constructor. Casting the object returned from `getObject()` to `Employee` will now work. Behind the scenes, the `DelegatorFactory` simply adds this interface to the set of those to be implemented by the dynamic proxy. Note that `DelegatorFactory` takes this interface object on trust: there is no validation that the interface doesn't introduce any new methods that are not already present in the interfaces of the superclasses.

Initializing the Superclasses

In “regular” Java, if a superclass does not provide a no-arg constructor, it’s necessary for the subclass to correctly initialize the superclass using constructor chaining. Normally this is done by including the superclass’s constructor’s argument(s) in the subclass’s constructor’s argument(s), and then passing them up the class hierarchy using `super()`.

The facilities shown in `DelegatorFactory` thus far do not support this. The `DelegatorFactory` is given a list of superclasses, and then instantiates an instance of each (to delegate to) using `java.lang.Class.newInstance()`. This requires a public no-arg constructor to exist.

If the would-be superclass does not offer a public no-arg constructor, the `DelegatorFactory` should be instantiated using a different constructor that takes preinstantiated superclass instances:

```
Person person = new Person("joe", 28);
Employment employment =
```

```
new Employment(someCalendar.getTime(),
               30000);
Employee employee =
    (Employee)
    new DelegatorFactory(
        new Object[] {
            person, employment
        },
        Employee.class
    ).getObject();
```

If the would-be superclass does not have a public constructor, or is abstract, a custom subclass (probably an anonymous inner class) should be instantiated and used instead.

Dealing with Diamonds

Typically, multiple implementation inheritance is used when the superclasses have orthogonal concerns. Certainly this is the case with `PersonLike` and `EmployeeLike`, and each method is unambiguous as to which ancestor it relates to.

However, sometimes there may be a common super-interface in the interfaces implemented by the “superclasses.” For example, suppose we have the concrete class, `Car`, which implements `Driveable`, the `Boat` class, which implements `Sailable`, and both `Driveable` and `Sailable` extend from `Steerable`. Since we want to use both `Car` and `Boat` to define a new subclass, we will also introduce a convenience interface, `AmphibiousCar` (see Figure 3).

The `steer()` method of `Steerable` is used to alter the bearing (0 to 359 degrees) of the steerable object. The `getBearing()` method, of course, should return this bearing.

For simplicity, the `drive()` method of `Driveable` and the `sail()` method of `Sailable` return a suitable string indicating the current bearing. Invoking `drive()` might return a string such as:

```
driving at bearing 30 degrees.
```

From what we currently know, we would create an amphibious car object using:

```
AmphibiousCar ac =
    (AmphibiousCar)
    new DelegatorFactory(
        new Object[] {
            Car.class, Boat.class
        },
        AmphibiousCar.class
    ).getObject();
```

What happens if we invoke the `steer()` method on our new amphibious car `ac`? Should the invocation handler delegate to the `Car` superclass object or

the `Boat`? The default behavior is to delegate to the first matching object. Hence, we will get:

```
ac.steer(30);
System.out.println(ac.drive());
// prints "driving at bearing 30 degrees"
System.out.println(ac.sail());
// prints "sailing at bearing 0 degrees"
```

The `Boat` superclass component of our class never knew that the bearing had changed.

It’s this kind of problem that persuaded the Java language designers to exclude multiple implementation inheritance. This is too large an area to cover in this article, but what we have here is an example of part of the so-called “diamond” problem, where there is a common ancestor. You can see the diamond in the interfaces: `Steerable`, `Driveable`, `Sailable`, and `AmphibiousCar`.

The `DelegatorFactory` utility deals with the diamond problem by allowing you to specify the invocation behavior to the delegate superclasses as a pluggable strategy (an example of the Strategy pattern). The strategy is defined by the `InvocationStrategy` interface. The default strategy (`InvokeFirstOnlyStrategy`) is to invoke the first ancestor superclass that can handle the method. However, in the case of the diamond, what is required is that both ancestors need to handle the method. The `InvokeAllStrategy` handles this. If the method being invoked has a nonvoid return type, the return value from the first ancestor is returned. The two strategies are shown in Figure 4.

The invocation strategy can either be set after the `DelegatorFactory` has been instantiated, or can be set through (yet another) overloaded constructor. Hence our amphibious car should be created using:

```
AmphibiousCar ac =
    (AmphibiousCar)
    new DelegatorFactory(
        Class[] {
            Car.class, Boat.class
        },
        new InvokeAllStrategy()
    ).getObject();
```

This time, we get:

```
ac.steer(30);
System.out.println(ac.drive());
// prints "driving at bearing 30 degrees"
System.out.println(ac.sail());
// prints "sailing at bearing 30 degrees"
```

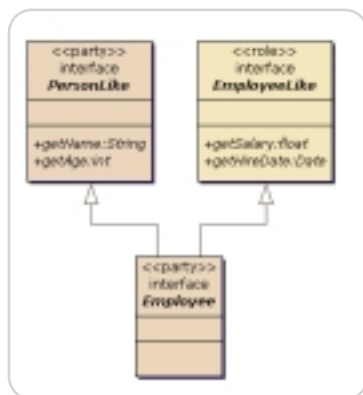


Figure 2 Employee unifies the two interfaces

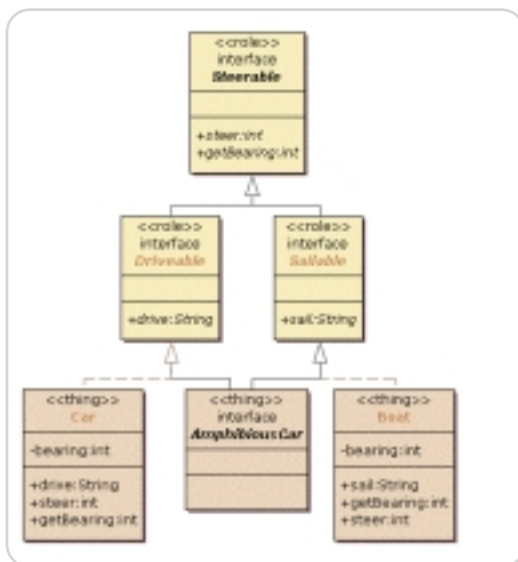


Figure 3 The AmphibiousCar interface creates a diamond

The `InvokeFirstOnlyStrategy` and `InvokeAllStrategy` are not the only strategies available (indeed we shall see one more shortly); however, they should work for most situations.

If a custom invocation strategy is required, it can be written by implementing the `InvocationStrategy` interface:

```
public interface InvocationStrategy {
    Object invoke(final List ancestors,
                 final Method method,
                 final Object[] args)
        throws Throwable
}
```

The `ancestors` parameter is an immutable list of the object instances representing the superclass. The `method` parameter represents the `Method` being invoked, and the `args` parameter contains the arguments to that `Method`. A typical invocation strategy would likely call `method.invoke(...)` somewhere within its implementation, with the first argument (the object upon which to invoke the method) being one of the ancestors.

We shall look at some applications of custom invocation strategies shortly. For now, though, an adaptation of `InvokeAllStrategy` might be to return the average return value of all ancestors, not just the return value of the first one.

Implicit Diamonds

In the previous diamond example, the `Steerable` interface is explicitly a super-interface of both `Driveable` and `Sailable`. What if the super-interface has not been explicitly factored out, though?

For example, in the original `PersonLike` and `EmployeeLike` example, what if each provided a `foo()` method, returning a string. Not imaginative, but never mind. Let's construct our employee and use an `InvokeAllStrategy`:

```
Employee employee = (Employee)
    new DelegatorFactory(new
Class[] { Person.class, Employment.class,
          Employee.class,
          new
InvokeAllStrategy()
    }.getObject());
```

Now let us invoke `foo()`:

```
employee.foo(); // what will happen?
```

Should the `Person`'s implementation be called, that of `Employment`, or both? Although you might wish that both

would be called (by virtue of our installed strategy), the sad truth is that only `Person`'s implementation would be called. This is because the dynamic proxy has no way of knowing which interface to match `foo()` to, so it simply matches it to the first interface listed. (It's a `java.lang.reflect.Method` that is passed to the `DelegatorFactory`, not the string literal "foo()". Methods are associated with a specific declaring class/interface.) In terms of the `DelegatorFactory`'s implementation, this means the first superclass listed in its constructor.

Note also that the compile time type does not matter. Neither of the following will change the outcome:

```
((PersonLike) employee).foo();
((EmployeeLike) employee).foo();
```

In fact, it would be possible to modify `DelegatorFactory` to make `InvokeAllStrategy` effective in this case, but that would involve parsing on the `Method.getName()` rather than the `method`. However, this has deliberately not been done. We'd rather you factored out the super-interface and made the diamond explicit. In the above example, add a `FooLike` (or `Fooable`) interface and make both `PersonLike` and `EmployeeLike` extend from it.

Other Applications

The issue raised by diamonds (implicit or otherwise) is that of how to deal with more than one implementation of a given method within an interface. However, it's interesting to turn this on its head.

In aircraft and other safety-critical environments, it's common to implement subsystems in triplicate. For example, there may be three different navigational systems, possibly with each implemented by different subcontractors. Each of these would be able to respond to the request, "Where is the location of the aircraft?"

Other systems within the aircraft interact with the navigational subsystem through a broker. This accepts the request on behalf of the navigational subsystem, and then forwards the request onto each implementation. Assuming there are no bugs in any of those implementations, they should all respond with the same data (within some delta of acceptable variance).

If there is a bug in one of the implementations, it may produce a response that differs wildly from the other two implementations. In this case, the broker disregards that response completely

and uses the responses of the other implementations that agree with each other.

The design of `DelegatorFactory` and its pluggable invocation strategies make it easy to implement such a broker. Imagine a `Calculator` interface that defines a single method `add(int, int):int`. We can then have three implementations of this interface, as shown in Figure 5.

The `FastCalculator` uses regular integer arithmetic. The `OneByOneCalculator` rather long-windedly performs its arithmetic by incrementing the first operand one-by-one in a loop. Both of these implementations are correct, just different. The final `BrokenCalculator` is just that; it actually performs a subtraction, not an addition.

The `InvokeSafelyStrategy` invocation strategy requires at least three ancestors that implement each method invoked. It will invoke the method on all ancestors, and then look to see that there is precisely one response that is most popular. Here is how to create a safe calculator that will ignore the incorrect implementation within the `BrokenCalculator`:

```
DelegatorFactory dfInvokeSafely =
    new DelegatorFactory(
        new Class[] {
            BrokenCalculator.class,
            OneByOneCalculator.class,
            FastCalculator.class
        },
        Calculator.class,
        new InvokeSafelyStrategy()
    );
Calculator safeCalculator =
    (Calculator) dfInvokeSafely.getObject();
assertEquals(7, safeCalculator.add(3, 4));
```

Note that the `InvokeSafelyStrategy` is not all that intelligent. It stores the return values from each ancestor within a `HashSet`, so it relies on an accurate implementation of `equals()` and `hashCode()`. If the actual return type were a float (wrapped within a `Float` object), a more sophisticated invocation strategy would most likely be required. In general, this strategy will work only with well-defined value objects that can intrinsically deal with any rounding and other such errors.

You could easily adapt or refine the `InvokeSafelyStrategy` into further strategies. For example:

- A parameterized version of `InvokeSafelyStrategy` could be used to deal with floats and other return types that would have rounding issues.

Dan Haywood has worked on numerous software development projects for more than 13 years. He's an independent consultant, trainer, and technical writer, having previously been a consultant for Sybase Professional Services and Accenture. His books include *Better Software Faster*, addressing the effective use and customization of Together Control Center, and the EJB chapters for *SAMS Teach Yourself J2EE in 21 Days*. Dan's latest interest also uses Java reflection heavily, namely Naked Objects (www.nakedobjects.org).

dan@haywood-associates.co.uk

- A background strategy might perform each invocation within a separate thread. Any invocation that had not responded within a certain timeout would be discarded.
- A high-performance system, on the other hand, might use a strategy that again uses a backgrounding strategy but returns the result of the first one to finish, killing off the rest.
- A logging strategy might perform some logging and then forward the invocation (typically to a single delegate).
- A caching strategy would check its cache with respect to the input parameter, and only if the result is unknown would it invoke the delegate (caching the subsequent result).
- A listener/broadcast strategy could represent a collection of listener objects; notifying all listeners of an event would require notifying only the broadcaster, which would then iterate over all listener objects as required.

Moreover, there is nothing to prevent multiple invocations from being chained together, (that is, the Decorator pattern). Alternatively, we could imagine a composite strategy (the Composite pattern) that combines a set of strategies together. Either the invocation chain (decorator) or the set of leaf strategies (composite) could be changed at runtime, meaning that we

can change the behavior and responsibilities of the object dynamically. This is a fundamentally different paradigm from conventional Java with its static typing. Normally, it's the type/class of the object that determines its behavior, something that cannot be changed once the object is instantiated. Here, though, we have ended up configuring the behavior of objects on an instance-by-instance basis: so-called instance-based programming. In effect, the whole notion of type becomes much less important.

There are echoes here too of aspect-oriented programming. Most aspect-oriented programming uses compile-time techniques (the term used is "weaving") to add in behavior to classes. The classic example of aspect-oriented programming is to add logging within all method calls. You can easily see, though, that these same features can be incorporated dynamically using invocation strategies; the decorator/composite invocation strategies would allow an arbitrary set of aspects to be added to a class. The difference though is that now the aspects are applied at runtime (and hence can be changed without recompile and redeployment).

Conclusion

The DelegatorFactory is simple to use, supporting classic mix-in (orthogonal) multiple-implementation inheritance "out-of-the-box" and – with its pluggable invocation strategy design – allows diamond hierarchies to be easily supported. Moreover, the design also lends itself to other quite unrelated problem spaces; for example, creating safe systems was explored. Taken to its logical conclusion, the approach supports both instance-based programming and aspect-oriented programming.

Of course, what makes DelegatorFactory work is Java's support for dynamic proxies, and that in turn requires that the ancestor superclasses implement interfaces. This approach won't work for class-based designs (JDOM is an example that comes to mind). But arguably class-based designs should be used only for value objects that should be final anyway. Those situations where multiple inheritance is desired are more likely to occur when working with reference objects.

One particular case deliberately not supported by DelegatorFactory is when there is a so-called implicit diamond. The solution though is to pull out the methods that appear in both interfaces, and move them into a new super-inter-

face. Then, make sure you use InvokeAllStrategy rather than the default InvokeFirstOnlyStrategy.

Of course, using a dynamic proxy object will be slower than a hand-crafted solution, principally because reflection is used. However, the difference may not be noticeable in practice. In recent releases of Java, Sun has put much effort in speeding up reflective invocation; as of JDK 1.4.1, it may well be that regular invocation is only twice as fast as reflective invocation (previously this figure was something like 40 times faster).

Using DelegatorFactory

The DelegatorFactory utility class and supporting classes described here can be downloaded from www.sys-con.com/java/sourcecode.cfm, and are compilable using Ant (v1.5.1 was used to create the build file). A JUnit-based test harness is also provided; JUnit v3.8.1 is required. The motivating examples in this article are based on the JUnit tests, so they should be easy enough to follow.

To run the tests with JUnit's text-based test runner, use:

```
ant test
```

Alternatively, you can use JUnit's test runner by running directly:

```
ant rebuild
java -classpath dist/halware-util-dynamic-bin.jar;dist/halware-util-dynamic-bin-test.jar
com.halware.util.dynamics.test.AllTests_gui
```

(The GUI test runner is not the default since JUnit's classloaders do not understand the Class-Path manifest attribute.)

I hope you find DelegatorFactory useful. It has been distributed under the GNU Lesser Public License, so you are free to embed it within your own software as required.

Acknowledgments

The inspiration for this article came from a session presented by Benedict Heal at the Object Technology Conference OT2002, run by the British Computer Society and the IEE. See www.ot2002.org/programme.html. Thanks, Benedict, for your further review comments on the draft of this article.

The UML class diagrams were created directly from the Java source code using Together ControlCenter, see www.borland.com.

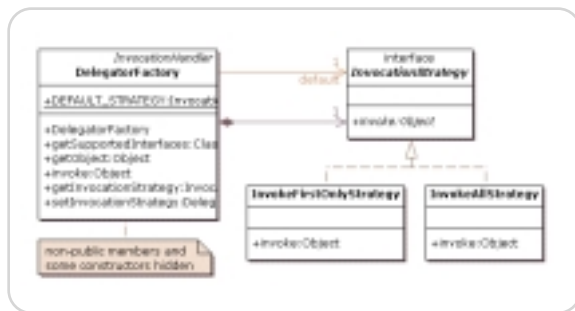


Figure 4 InvocationStrategies are pluggable

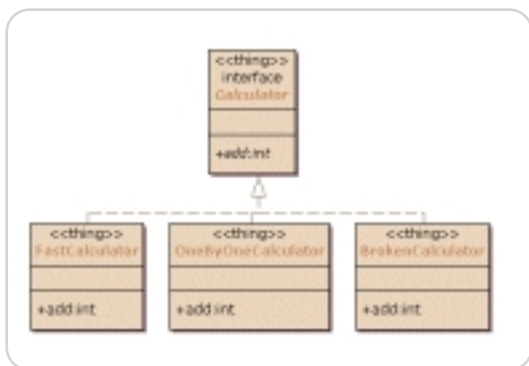


Figure 5 Three implementations of a Calculator

SYS-CON MEDIA

304,187 of the World's Foremost IT Professionals

DIRECT MAIL, EMAIL OR MULTI-CHANNEL

Target CTOs, CIOs and CXO-level IT professionals and developers who subscribe to SYS-CON Media's industry leading publications

Java Developer's Journal...
The leading publication aimed specifically at corporate and independent java development professionals



PowerBuilder Developer's Journal...
The only PowerBuilder resource for corporate and independent enterprise client/server and web developers



ColdFusion Developer's Journal...
The only publication dedicated to ColdFusion web development

LinuxWorld Magazine...
The premier monthly resource of Linux news for executives with key buying influences



WebLogic Developer's Journal...
The official magazine for BEA WebLogic application server software developers, IT management & users

Web Services Journal...
The only Web Services magazine for CIOs, tech, marketing & product managers, VARs/ISVs, enterprise/app architects & developers



.NET Developer's Journal...
The must read iTechnology publication for Windows developers & CXO management professionals

XML-Journal...
The world's #1 leading XML resource for CEOs, CTOs, technology solution architects, product managers, programmers and developers



WebSphere Developer's Journal...
The premier publication for those who design, build, customize, deploy, or administer IBM's WebSphere suite of Web Services software



Wireless Business & Technology...
The wireless magazine for key corporate & engineering managers, and other executives who purchase communications products/services

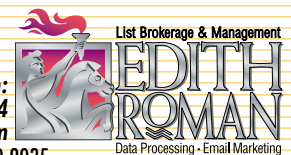
Recommended for a variety of offers including Java, Internet, enterprise computing, e-business applications, training, software, hardware, data back up and storage, business applications, subscriptions, financial services, high ticket gifts and much more.

NOW AVAILABLE!
The SYS-CON
Media Database
304,187 postal
addresses



For email information:
contact Frank at 845-731-3832
frank.cipolla@epostdirect.com
epostdirect.com 800-409-4443 fax 845-620-9035

For postal information:
contact Kevin at 845-731-2684
kevin.cology@edithroman.com
edithroman.com 800-223-2194 fax 845-620-9035



From Within the Java Community Process Program

'Tigers' to MIDPs



Onno Kluyt

This month I'll discuss the evolution of the JCP, J2SE 1.5 or "Tiger", Java portlets, and a new JSR from Nokia and Siemens.

JSR 215, aka JCP Version 2.6

The Java Community Process is the only standards body with a version number! Currently, we're at 2.5 and hope to soon be at 2.6. Where are we now? About halfway between 2.5 and 2.6. Rule and regulation changes in the JCP happen through the JSRs. JSR 913 modified the JSR ballot voting rules, JSRs 99 and 171 led to JCP 2.5, and JSR 215 is creating JCP 2.6. This JSR just completed the Community Review and ballot. The Program Office together with the Executive Committees will now be working toward Public Review. A few of the things the group will focus on are clarifications around JSRs assigned to both ECs, a draft transparency plan for spec leads to use, and ironing out the mandatory TCK requirements. While the Community Review period has passed, the draft is still available on the Web site and you can send in your thoughts and ideas. Speaking of evolution, at the end of this year the JCP will be five years old. The Program Office will be at ApacheCon in November to celebrate. Now, on to the real work in the community!

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

onno@jcp.org

A Tiger in Review

The three main Java platforms (J2ME, J2SE, and J2EE) are all done through the JCP. The coordination for

these main releases takes place through so-called Umbrella JSRs. The actual API work for a new version of J2SE or J2EE does not happen in the Umbrella JSR. Instead the Umbrella JSR references the individual JSRs that specify new and updated APIs. Many of the JSRs that contribute to J2SE 1.5 or "Tiger" have just completed their Community Reviews. See JSR 176, the J2SE 1.5 Umbrella JSR, for a complete list of component JSRs. I covered a few of these JSRs in previous columns. Here I'd like to mention JSRs 3, 13, 199, 204, and 206. With "Tiger" the Java Management Extension specification becomes part of the J2SE distribution. JSR 13 adds floating point arithmetic to BigDecimal so that decimal numbers can be used for general purpose arithmetic without the need to convert to and from other types. The Java Compiler API enables a Java program to invoke a Java language compiler programmatically. JSR 204 further enhances the internationalization capabilities of the Java platform by providing support for the Unicode 3.1 standard. Unicode 3.1 defines characters that cannot be described by single 16-bit code points. Finally, there is JSR 206, which is developing JAXP version 1.3, an API for processing XML.

The Java Portlet Specification

This JSR, number 168, is co-led by IBM and Sun. The JSR resulted from a simultaneous submission of two quite similar JSRs individually presented by

both companies. At the urging of the EC, IBM and Sun withdrew those individual JSRs and submitted a combined one, JSR 168. The JSR recently posted in short succession two Proposed Final Drafts, and it's very likely that by the time you read this column the JSR will be on the Final Approval Ballot. This specification builds on the servlet technology by defining the desktop metaphor for the aggregation of servlets and JSPs. It also covers security and personalization, and enables interoperability between portlets and portals.

JSR 228

Nokia and Siemens recently finalized JSR 195, Information Module Profile. This was quickly followed by the submission of JSR 228 that defines Information Module Profile - Next Generation. The technology is aimed at devices that want to support a MIDP 2.0 environment but don't provide any graphical display capabilities required by MIDP 2.0. JSR 195 first opened this market for Java-enabled devices, such as modems, metering, and home electronics. This created a strong desire for the advanced capabilities of MIDP 2.0. JSR 228 will focus on the domain security model, HTTPS and secure networking, OTA provisioning, and push architecture. The spec leads aim to finish the JSR in the late spring of 2004.

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

Industry News

Wily Launches Study on Java Application Performance

(Brisbane, CA) - Wily Technology has launched a comprehensive survey of J2EE application performance. The 2003 Wily Benchmark Survey of J2EE Application Performance and Availability will measure industry-wide user experiences with J2EE application deployment and management across a spectrum of enterprise and public sector organizations worldwide.

The survey report will be published in November 2003. Participation in the

survey is open to qualified respondents active in the design, development, deployment, and management of J2EE applications within an organization. www.wilytech.com/2003/chartingJ2EE

Sun Shrinks 225 Java Products to Six Systems

(San Francisco) - Sun Microsystems has launched its third quarterly set of coordinated product releases, combining 225 products into just six systems.

As part of its push toward reducing cost and complexity, Sun will in the

future deliver systems around six Java-branded software products designed to integrate all needed applications and services on server, desktop, development, and operational environments. A new simplified charging approach, priced per employee, was also introduced.

The six Sun systems are the Java Enterprise System, the Java Desktop System, Java Enterprise Studio, and N1, and for future deployment - the Sun Java Mobility System and Sun Java Card System.

www.sun.com ☺

Advertiser	URL	Phone	Page
Altova	www.altova.com		19
Bea dev2dev	bea.com/dev2devdays2003	1-925-287-5156	29
Borland Software Corporation	go.borland.com/j6		11
Borland Conference 2003	connect.borland.com/borcon03		37
Canoo Engineering AG	www.canoo.com/ulc/	41 61 228 9444	7
COMDEX Las Vegas 2003	COMDEX.com		43
Computer Associates	ca.com/lifecycle		4
Crystal Decisions	www.crystaldecisions.com/lbl/	1-800-877-2340	9
Edith Roman	www.edithroman.com	800 223 2194	63
GreenPoint, Inc.	www.webcharts3d.com/demo		47
Extentech	www.extentech.com/jdj	415-759-5292	35
IBM Rational	ibm.com/rational/seamless		21
InetSoft Technology Corp.	www.inetsoft.com/jdj	888-216-2353	33
Infragistics, Inc.	www.infragistics.com	800-231-8588	14-15
INT, inc.	www.int.com	713-975-7434	48
iSavix	http://isavix.net	703-689-3190	39
LinuxWorld Magazine	www.linuxworld.com	888 303 5282	57
MX Developer's Journal	www.sys-con.com/mx/subscription.cfm	888 303 5282	55
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	44
Oak Grove Systems	www.oakgrovesystems.com/jdj	818-440-1234	31
Parasoft Corporation	www.parasoft.com/jdj10	888-305-0041	Cover II
Parasoft Corporation	www.parasoft.com/jdj10	888-305-0041	23
QUALCOMM Incorporated	www.qualcomm.com/brew/jdj		51
Quest Software, Inc.	http://java.quest.com/performance/jdj		17
Quest Software, Inc.	http://java.quest.com/jclass/jdj		27
Quest Software, Inc.	http://java.quest.com/jprobe/jdj		Cover IV
ReportingEngines	www.reportingengines.com/info/trial3.jsp	888-884-8665	28
SAP	www.sap.com/netweaver	1-800-880-1727	13
Software FX	www.softwarefx.com	800-392-4278	Cover III
SYS-CON Events	www.sys-con.com/edgeeast2004	201 802 3069	45
SYS-CON Media	www.sys-con.com/2001/sub.cfm	888 303 5282	53
WebAppCabaret	www.webappcabaret.com/jdj.jsp		25
Zero G	www.zerog.com	415-512-7771	3

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *Java Developer's Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in *Java Developer's Journal*. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.

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

Next Month

Using Java Generics

Have you heard? Generics will be in the next release of the Java SDK (code named Tiger aka JDK 1.5). You might be wondering "What is a generic?" or "Why should I care?" or even "Cool! How do I use them?" No matter what your level of interest, this article will introduce generic coding, explain how they are used and what their advantages are, and discuss how they will impact your work.

Life Outside the Sphere

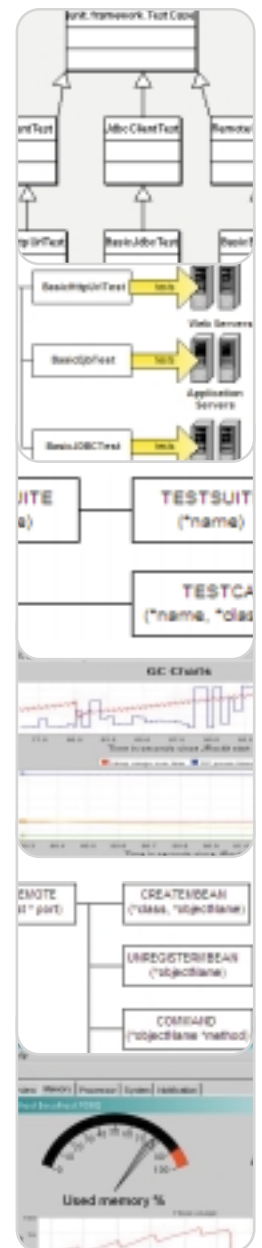
IBM's WebSphere Device Developer (WSD) is a sophisticated development platform for WebSphere Micro Environment (WME, also known as J9). Debugging, profiling, packaging - whatever you want, WSD can do it all. Based on Eclipse, it's just right for those of us who like to work with Eclipse. The problems start if you prefer to use some other IDE or you believe in automated, continuous integration. This article will show you how to master using WME without WSD.

Managing J2EE Systems with JMX and JUnit

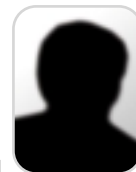
The promise of J2EE was to build more robust, scalable, and secure enterprise systems quickly and easily. J2EE is supposed to take the complexity out of building powerful distributed systems. But as with the J2EE spec, these systems usually suffer through management only as an afterthought. This article will show you how to use the open source JMX4ODP framework to combine the ease of JUnit tests and the extensible management of JMX beans.

WebLogic JRockit 8.1 by BEA Systems

The JRockit engineers made two assumptions when they first designed JRockit. First, server VMs run for a long time and, second, memory is cheap and plentiful. This motto still rings true in BEA's offering of the 8.1 (J2SE 1.4.1_03) version of this product. And, unlike the more familiar JVMs, this VM comes with a face.



One IDE to Rule Them All



Henry Roswell

At JavaOne, Jonathan Schwartz, executive vice president of Sun's Software Group, outlined his mission to increase the number of Java developers from 3 million to 10 million. The hope is to attract these extra seven million from the legions of Visual Basic (VB) developers. Visual Basic's strength comes from a tool experience that is inseparable from the language and, in order to capture their mind share, Java needs the killer IDE.

Early Java programming books were often bundled with a copy of Visual Café, allowing readers to concentrate on learning the language syntax instead of esoteric JDK command syntax. While some programmers pride themselves on writing macros to customize their favorite text editors, integrated development environments (IDEs) offer the easy life of code assist, incremental compilation, and integrated debuggers.

Since Visual Café, a number of great IDEs have been created. JBuilder, IntelliJ, and Eclipse lead the 2003 *DDJ* Readers' Choice Awards with TogetherJ, Oracle9i Developer, and Sun ONE all equally worthy of pole positions. The irony is that having so many good development tools is a weakness, not a strength, when it comes to tackling Microsoft.

The current Java IDE landscape makes extensibility APIs either constrained to the lowest common denominator or proprietary to each vendor.

JavaBeans and JavaServer Faces (JSF) are examples of the former because, while components can be good citizens for how a tool will use them, they cannot exercise sufficient control over the environment hosting them. To truly leverage a tool requires knowledge of its object model for representing artifacts, the life-cycle API for how data is persisted, and the control of event notification between viewers. If the IDE surfaces these internals to the component, a much richer edit time experience can be created. The case in point is Microsoft Design-Time Control (DTC), which allows customization of Web page components through in-place ActiveX controls that run within the source editor. Java's answer to DTC is JSF. Without being able to surface a satisfactory mechanism to plug into the IDE's viewers, the source-editing experience relies on using the JSF component as you would any other JavaServer Page (JSP) tag library or JavaBean. This is not going to lure the VB crowd who want in-place preferences dialogs for their component embedded directly in the source page.

The most successful challenge to Microsoft tools in the Java space so far is to adopt a proprietary approach as used by JBuilder, IntelliJ, or Eclipse. These surface the APIs that tool vendors and component builders need to leverage the edit time experience. However, their bespoke

interfaces cause fragmentation in the tools space, and while JSR 198 is a well-intentioned attempt to resolve this problem, it's too little too late and is fated to become the lowest common denominator.

Any successful extension API needs to be more than skin deep, and what motivation do the tools vendors have to come together and agree on a common object model or life-cycle API? It is IBM's doomed AD cycle all over again. If a compromise API is reached, the IDE vendors will do half-hearted shoe horn-ing of this into their existing tool, while still retaining their internal extension APIs for serious platform development. The issue is further complicated by the inability of Swing and SWT to interoperate, and if the GUI toolkit can't be agreed upon, there is surely little hope that the internals can converge.

The only solution I see is for one of the existing IDEs to become the de facto tool for Java. The benefit of having only one tool is that people can program to its extension API, have access to the internals of its object model and construction and, in an ideal scenario, the tool would be offered with JDK downloads to round off the whole Java "out of the box" experience. This way when the seven million newcomers we are hoping to attract first taste Java, they feel at home with a rich set of design-time tools fully integrated with the language. ☺

Henry Roswell is a veteran consultant who would like to think he's seen it all, but is constantly amazed by new events everyday.

henry@sys-con.com

Java Dudes

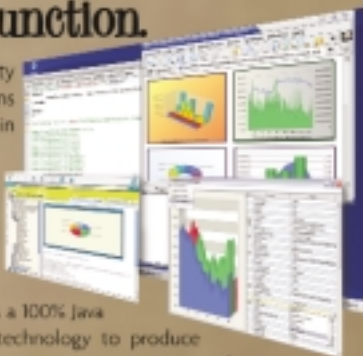


A Fresh Brew of Chart FX...



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 Programmer's Guide, the Javadoc API and hundreds of samples. *For more information or to download a trial version, visit www.softwarefx.com.*



Chart FX
for JAVA

www.softwarefx.com

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

Introducing JProbe® 5.0

...now smarter and faster than ever

JProbe®

Find the cause of J2EE code performance, memory and threading problems faster than ever before with JProbe 5.0. 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



For more info and a free eval, visit:

<http://java.quest.com/jprobe/jdj>