

No. 1 *i*-Technology Magazine in the World

JDJ

SEPTEMBER 2004 VOLUME:9 ISSUE:9

The Java
Job Market

PAGE 10

WHEN MARS IS TOO BIG TO DOWNLOAD



PLUS...

- ▶ Connecting the Java World to Grid-Enabled Databases
- ▶ Writing Java Card Applications
- ▶ Beyond Entity Objects

RETAILERS PLEASE DISPLAY UNTIL OCTOBER 31, 2004

\$9.99US \$9.99CAN

10>



0 74470 01751 6

The new stud on the server farm.



Presenting the new Xserve® G5, a wickedly fast, extremely compatible and refreshingly affordable 1U server from Apple.

With dual 2GHz 64-bit G5 processors, it achieves blazing speeds of up to 30 gigaflops. It's so powerful, in fact, that the U.S. government is deploying* 1,566 Xserve G5 servers to create one of the world's fastest supercomputers, capable of up to 25 trillion calculations per second.

And it comes complete with Mac OS® X Server, Apple's UNIX-based operating system that provides a complete suite of standards-based network services with no per-client fees. So whether you have Mac, Windows, UNIX or Linux clients, Xserve is ideal for cross-platform file sharing, hosting dynamic websites, streaming audio and video and running powerful J2EE applications – right out of the box.

Of course, its most impressive feature may be its price, starting at just \$2,999†. The new Xserve G5.





WILL IT WORK?
Should you even
have to ask?

Supports Oracle 10g

Don't take chances connecting your application to your data. Rely on DataDirect Technologies™ for premium JDBC drivers with support for advanced functionality like distributed transactions, connection pooling, and BLOB/CLOB. Our Type 4 drivers are fully database independent and are the SPECjAppServer/ECPerf performance and scalability leader.

Download your free evaluation today. www.datadirect.com/jdj

DataDirect[™]
TECHNOLOGIES
www.datadirect.com 800-876-3101

JDJ contents

JDJ Cover Story

Generating simple terrains with Java3D

When Mars Is Too Big to Download

by Michael Jacobs

52

FROM THE GROUP PUBLISHER
Wanted: 19 More of the Top Software People in the World
 by Jeremy Geelan **6**

VIEWPOINT
The Evolution of Web Application Technologies for Java
 by Craig R. McClanahan **8**

JAVA ENTERPRISE VIEWPOINT
Interviewing Enterprise Java Developers
 by Yakov Fain **10**

COMMUNICATIONS
JAIN/SLEE
Opening the telecommunications world for Java
 by Sven Haiges **12**

JDBC
Connecting the Java World to Grid-Enabled Databases
Consolidate IT resources and optimize usage
 by Kuassi Mensah **26**

CORE AND INTERNALS VIEWPOINT
Three Gems from JavaOne
 by Calvin Austin **34**

SOLUTIONS
Java GoF Creational Design Patterns
For cleaner development and easier maintenance
 by Puneet Sangal **36**

SMART CARDS
Writing Java Card Applications
Writing applications for the smallest JVM
 by Vijay Phagura and Anita Phagura **40**

FIRST LOOK
Building Applications with Berkeley DB Java Edition
High-performance database goes pure Java
 by Jim Menard **46**

DESKTOP JAVA VIEWPOINT
Turkish Java Needs Special Brewing
 by Joe Winchester **50**

LABS
JDeveloper 10g
 by Oracle Corporation
 Reviewed by Alain Trottier **58**

JSR WATCH
From Within the Java Community Process Program
Updating the first JSR
 by Onno Kluyt **60**

@ THE BACK PAGE
One Man's Open Source, Another Man's Asset
 by Henry Roswell **62**

Feature

20



Beyond Entity Objects

by Bill Kohl

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for \$69.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. Postmaster: Send address changes to: JDJ, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.



Jeremy Geelan

Wanted: 19 More of the Top Software People in the World

For over a decade, Tim Bray, one of the prime movers of XML, managed the Oxford English Dictionary project at the University of Waterloo. That was from 1988 to 1999. During the end of his time there he launched one of the first public Web search engines (in 1995), coined XML 1.0, and coedited “Namespaces in XML” (1996–1999).

Bray is therefore no technological slouchabout. Nor is he without deep insight into the ways of the Web, having served as a Tim Berners-Lee appointee on the W3C Technical Architecture Group in 2002–2004, after which he joined Sun as director of Web technologies in March of this year. So when he takes the trouble to describe someone as “probably one of the top 20 software people in the world,” you know he means it.

The person in question was Adam Bosworth, famous for Quattro Pro, Microsoft Access, and Internet Explorer 4 even before he joined BEA as VP of engineering in 2001, when BEA bought Crossgain, the company he’d by then cofounded after leaving Microsoft. He went on to become BEA’s chief architect before, very recently, leaving the Java app server company to join Google, Inc.

Bray was one of the gurus that a headhunter working with Google, Inc., called for a reference before they hired Bosworth. Bray gave him a glowing one. That’s when Bray’s description of him as probably one of the top 20 software people on earth appeared. As we all know, Bosworth got the job and now works on software that is very different from what he was architecting at BEA.

“Rather than worrying about what the IT of large corporations needs to do to support the corporation,” he explained recently, “I’m worrying about mere mortals. In fact, my mom.”

Bosworth says he can only build software if he first gets some mental image in his head of the customers. Who are they? How do they look, feel, think? He calls this “designing by guilt,” which he explains as follows: “Because if you don’t do what

feels right for these customers, you feel guilty for having let them down.”

Of course, customers are endlessly disparate, complex, heterogenous, and distinct. But even so, Bosworth says he has always found it necessary to think about a small number of distinct types of customers, and then design for them. “And boy is it satisfying to do this when the people you are designing for are your friends, family, relatives, your smart-aleck son, and so on,” Bosworth observes, “and when even your mother can use what you build – I call this the mom factor. It’s corny but fun.”

What a refreshing approach. No wonder, with this high regard for technology’s fundamentals, Bray rates Bosworth as one of the top 20 software people in the world. The question naturally arises, however: who are the other 19?

This is not easy to answer, and not because there are too few candidates but because there are too many. In a phase of the economic cycle most readily remembered for being downbeat and understated, the names of leading *i*-technologists – whom Internet technologies rely on for their unceasing innovation and ingenuity – nonetheless still trip off most people’s lips. Just think of Sergey Brin, Bill Jay, Linus Torvalds, Tim Berners-Lee, James Gosling, Anders Hejlsberg, Don Box, Nathan Myhrvold, W. Daniel Hillis, Mitch Kapor...

The “technorati” or “digerati” – call them what you will – the aristocracy of the online world. Can a list of the Top 20 *i*-Technologists possibly be compiled that doesn’t cause the online equivalent of fistfights when published? Obviously not. But that shouldn’t deter us from trying. So, have at it. The final list will be reported here, along with the near-misses. You can send your nominations, including your reasoning, to *i*-Technology’s Top Twenty, toptwenty@sys-con.com. It will take more than a month to ensure that everyone with something worth saying has found time, energy, and above all the appropriately persuasive argument to persuade us of the merits of their choice/s. We’ll report next issue on how this process is going. ☺

Editorial Board

Desktop Java Editor: **Joe Winchester**
 Core and Internals Editor: **Calvin Austin**
 Contributing Editor: **Ajit Sagar**
 Contributing Editor: **Yakov Fain**
 Contributing Editor: **Bill Roth**
 Contributing Editor: **Bill Dudney**
 Contributing Editor: **Michael Yuan**
 Founding Editor: **Sean Rhody**

Production

Production Consultant: **Jim Morgan**
 Associate Art Director: **Tami Beatty-Lima**
 Executive Editor: **Nancy Valentine**
 Associate Editors: **Jamie Matusow**
Gail Schultz
 Assistant Editor: **Torrey Gaver**
 Online Editor: **Martin Wezdecki**
 Research Editor: **Bahadir Karuv, PhD**

Writers in This Issue

Calvin Austin, Yakov Fain, Jeremy Geelan, Sven Haiges, Mike Jacobs, Onno Kluyt, Michael Havey, Pramod Jain, Yayati Kasralikar, William Kohl, Craig R. McClanahan, Jim Menard, Kuassi Mensah, Anita Phagura, Vijay Phagura, Henry Roswell, Puneet Sangal, Alain Trottier, 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:

888 303-5282
 201 802-3012

subscribe@sys-con.com

Cover Price: \$5.99/issue. Domestic: \$69.99/yr. (12 Issues)
 Canada/Mexico: \$99.99/yr. Overseas: \$99.99/yr. (U.S. Banks or Money Orders) Back Issues: \$10/ea. International \$15/ea.

Editorial Offices

SYS-CON Media, 135 Chestnut Ridge Rd., Montvale, NJ 07645
 Telephone: 201 802-3000 Fax: 201 782-9638

Java Developer’s Journal (ISSN#1087-6944) is published monthly (12 times a year) for \$69.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. Postmaster: Send address changes to: Java Developer’s Journal, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

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

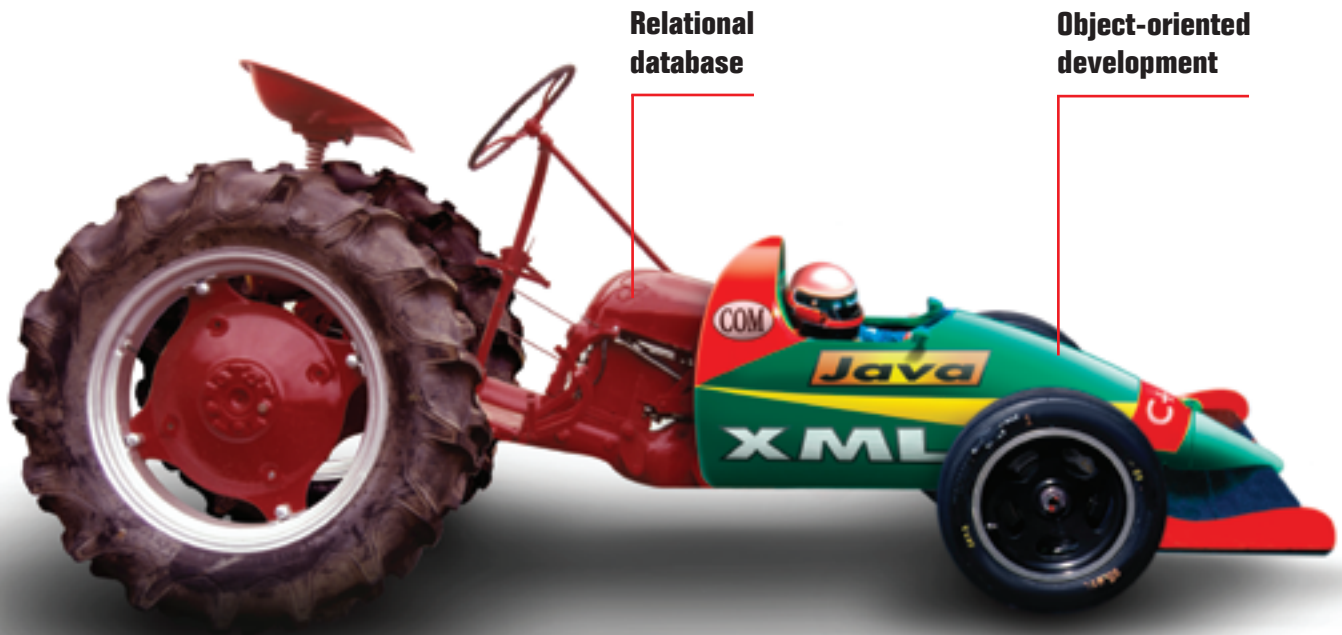
Newsstand Distribution Consultant
 Brian J. Gregory/Gregory Associates/W.R.D.S.
 732 607-9941, BJGAssociates@cs.com

©Copyright

Copyright © 2004 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission. For promotional reprints, contact reprint coordinator Kristin Kuhnle, kristin@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.

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.





TIME TO CHANGE YOUR DATABASE

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

Caché, the *post-relational* database from InterSystems, combines high-performance SQL for faster queries and an advanced object database for rapidly storing and accessing objects. With Caché, no mapping is required between object and relational views of data. Every Caché class can be automatically projected as Java classes or EJB components with bean-managed persistence. Plus, every object class is instantly accessible as tables via ODBC and JDBC.

That means huge savings in both development and processing time. Applications built on Caché

are massively scalable and lightning fast. They require little or no database administration. And Caché's powerful Web application development environment dramatically reduces the time to build and modify applications.

We are InterSystems, a specialist in data management technology for over twenty-six years. We provide 24x7 support to four million users in 88 countries. Caché powers enterprise applications in healthcare, financial services, government, and many other sectors. Caché is available for Windows, OpenVMS, Linux, and major UNIX platforms – and it is deployed on systems ranging from two to over 10,000 simultaneous users.



Try a better database. For free.

Download a free, fully-functional, non-expiring version of Caché or request it on CD at www.InterSystems.com/match1



Craig R. McClanahan

The Evolution of Web Application Technologies for Java

One of the primary values of the Java platform has been the concept of “write once, run anywhere.” A key factor in achieving this goal has been the fact that Java, as a platform, has focused on defining standardized API specifications, which can be implemented by multiple providers who can compete on features and performance. For the application developer, the key benefit is portability across these implementations, avoiding vendor lock-in – as long as the standardized APIs are of sufficient power and usability to allow developers to create applications based on the standards, without having to rely on vendor extensions.

The servlet API was the first “extension” API to be standardized. Its mission was to encapsulate the HTTP protocol in an object-oriented style accessible to Java developers, with added features that help maintain state across HTTP requests. The original design has proven to be remarkably resilient, with subsequent versions focusing on packaging (Web application archives), filters (chain of responsibility pattern), and event handling.

Servlets, however, are primarily attractive to developers who understand Java. In addition, Web applications built directly on top of the servlet API tended to intermix presentation logic – the generation of the HTML markup – with business logic, leading to applications that were difficult to maintain and enhance.

The first issue was addressed by the development of JavaServer Pages (JSP) technology. Instead of embedding HTML generation inside a Java servlet, JSP pages turn the servlet inside out – the source code for a JSP is primarily HTML markup, with special markup elements (in JSP, known as custom tag handlers) used to insert dynamic content. Dealing with the second issue, however, has led to a plethora of innovative approaches to the overall architecture of a Web application.

One particular design pattern – the Model-View-Controller (MVC) pattern

adapted from rich-client architectures – has proven particularly useful. The most popular implementation has been the Apache Struts Framework, an open source Web application framework that implements this design. It enables a separation of concerns between presentation logic and business logic, while providing useful additional functionality. The result has been its adoption as a de facto standard by nearly every major tool that supports Java Web application development.

While Struts has become the epitome of the overall architecture for Web applications, it does not include sophisticated components for the development of sophisticated user interfaces. The recent release of version 1.0 of the JavaServer Faces specification has addressed the need for standardized fundamental APIs for components, which will lead to the development of robust tools and interoperable component libraries.

What happens to a de facto standard when a portion of its functionality is later standardized? It depends on whether the existing technology embraces or ignores the newcomer. In the case of Struts, we’ll see an embracing of JavaServer Faces just as it embraced the JSP Standard Tag Library (JSTL) earlier. An integration library, already available as nightly builds, allows you to migrate an existing application, one page at a time, to use JavaServer Faces components in the presentation tier – without changing the business logic embedded in actions. In other words, the separation of concerns provided by the framework really works.

I have enjoyed my personal participation in the evolution of the Web tier APIs in Java. My initial participation was in an open source implementation of the servlet API, which became Tomcat. I was the initial creator of the Struts framework, and co-spec lead for version 1.0 of JavaServer Faces. In my current role (architect on the team building Sun Java Studio Creator), I can assure you that my contributions will continue to be felt. ☘

Craig McClanahan is a senior staff engineer at Sun Microsystems, Incorporated. Presently, he is an architect on the team building Sun Java Studio Creator, a GUI development tool for building Web applications. Previously, he was the original architect of the servlet container inside Tomcat 4.x and 5.x, principal author of the Struts Framework, and co-Specification Lead for JavaServer Faces 1.0.

craig.mcclanahan@sun.com

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 Mussa megan@sys-con.com
Associate Sales Managers:
Kristin Kuhnle kristin@sys-con.com
Beth Jones beth@sys-con.com
Dorothy Gil dorothy@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
Assistant Editor:
Torrey Gaver torrey@sys-con.com
Online Editor:
Martin Wezdecki martin@sys-con.com

Production

Production Consultant:
Jim Morgan jim@sys-con.com
Lead Designer:
Tami Beatty-Lima tami@sys-con.com
Art Director:
Alex Botero alex@sys-con.com
Associate Art Directors:
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Assistant Art Director:
Andrea Boden andrea@sys-con.com

Web Services

Vice President, Information Systems:
Robert Diamond robert@sys-con.com
Web Designers:
Stephen Kilmurray stephen@sys-con.com
Matthew Pollotta matthew@sys-con.com

Accounting

Financial Analyst:
Joan LaRose joan@sys-con.com
Accounts Payable:
Betty White betty@sys-con.com
Account Receivable:
Shannon Rymza shannon@sys-con.com

SYS-CON Events

President, SYS-CON Events:
Grisha Davida grisha@sys-con.com
National Sales Manager:
Jim Hanchrow jimh@sys-con.com

Customer Relations

Circulation Service Coordinators:
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com
JDJ Store Manager:
Brunilda Staropoli bruni@sys-con.com



The right Java, whatever the gig.

Borland® JBuilder.® The #1 Java™ tool in the world for a reason. Pick the size that fits your needs. Automate the routine stuff. Handcraft the unique. Have an active voice at every stage in the process. Move faster, and make every project a hit. Whether your application is headed to the Web, enterprise, or mobile, just pick the feature set you need. And start rockin'!

Customizable code editor • Refactoring • Local and remote debugging • Integrated unit testing • Two-way visual Struts designer • JSP™ tag library/framework support • XML and Web Services • Mobile application development • Advanced build and configuration management with Apache® Ant • Visual EJB™ designer • Two-way deployment descriptor editor • Archive builder • Integration with all major J2EE™ application servers

go.borland.com/j6

Made in Borland® Copyright © 2004 Borland Software Corporation. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. • 21715-9

Borland®
Excellence Endures™

Learn from the experts at the premier event for technical education – the 2004 Borland Conference – <http://connect.borland.com/borcon04s>



Yakov Fain
Contributing Editor

Interviewing Enterprise Java Developers

Today's Java job market is healthy. Major online job search engines show thousands of openings, and people are competing for these jobs. Skilled Java developers are just as popular as Visual Basic or PowerBuilder developers were back in 1996. There is a major difference though – back then, client/server developers could make a decent living by mastering one front-end tool and any major relational DBMS. These days a Java developer has to know about 10 different tools or technologies to find a good job and feel relatively secure for a couple of years.

During the last year I've been interviewing lots of J2EE developers, who are in demand again. But over the last several years job requirements, people, and résumés of Java developers have changed quite a bit and this is what I've noticed:

- People do not call themselves Java developers or programmer-analysts anymore – most of them prefer the title of Java architect. Unfortunately, only some of them really understand how J2EE components operate and can suggest some design solutions.
- Job applicants are more senior and I barely see any college graduates or junior programmers in the market. Many of the junior positions are being outsourced and the number of graduates with computer science degrees has declined over the past several years.
- Java certification does not make your résumé stand out. Actually, if a résumé starts with a list of Java certifications, most likely it's a beginner. I'm not against certification as it helps you learn the language or a tool, and shows that you are willing and can study. But the fact that you have a Java certificate doesn't mean that you're a skilled professional.

- Three to four years ago people with EJB experience were in high demand; now Struts is a more valuable asset. This is a good framework for Web applications, but it has the following side effect: some Struts developers don't really know what's under the hood and how plain vanilla servlets work. When I ask how an HTML form is being processed by a servlet, they start from the class Action.
- On a similar note, some people don't know exactly how JDBC works – they just pass a SQL statement to some wrapper class created by local architects and get the result set back.
- I see a new breed of Java architects who used to be project managers. These people usually know their business really well, can talk about application servers, messaging and clusters, and capacity planning, but often fall short on Java technical questions.
- Job requirements are longer these days and recruiting companies don't even want to submit your résumé to the client if you have "only" 9 out of 10 required skills. As a matter of fact, recruiters screen candidates a lot better now.
- Be prepared to pass at least four interviews to get hired. While back in 1999 two good interviews would be enough, in 2001 it was very difficult to even get an interview let alone a job!

What does a good J2EE developer have to know in addition to understanding the difference between abstract classes and interfaces? Usually employers are looking for people with at least 10 of the following skills: Java servlets, JSP, Struts or a similar framework, EJB, JMS, any commercial message-oriented middleware, JDBC, JNDI, HTML, XML, Ant, SQL,

one of the major application servers, a couple of relational database management systems, any UML modeling tool, several design patterns (at least a Singleton!), and familiarity with Unix. Next year JavaServer Faces and Hibernate will most likely be included in this laundry list.

Understanding why a particular J2EE component is being used in your project is equally important. If the interviewer asks you, "Why did you use EJB in this project?" please do not answer, "This decision was made before I joined the project." Have your own opinion and explain why you think it was a good or bad choice for this particular project.

I keep hearing the "horror stories" about questions some people get during interviews. In my opinion, the interviewers should ask more open-ended questions about the applicant's prior experience, going into technical details when appropriate. I don't think it's fair to ask a person to write a Java program processing a binary tree or implementing a finite state machine. These are the things that can be looked up online or in the books when needed.

Good knowledge of the business terminology of your potential employer is also important. I'm not sure about the Silicon Valley or Europe, but here in New York just being a techie may not be good enough to get a senior job. For example, if you're applying for a Java position in a financial brokerage company and don't know what a short sale is, this may be a showstopper. If you are a senior developer, you should be able to hit the ground running... Try to find out from your recruiter as many details as possible about the business of your potential employer, do your homework, and you'll get the job! They are desperately looking for good Java people and you can be one of them. ☺

Yakov Fain works as a Java architect for Bank of America in New York City. He wrote the book *The Java Tutorial for the Real World*; an e-book *Java Programming for Kids, Parents and Grandparents*; and several chapters for the book *Java 2 Enterprise Edition 1.4 Bible*. Yakov holds a masters degree in applied mathematics. For more information please visit www.smartdataprocessing.com.

yakovfain@sys-con.com

We have a problem.

J2EE application problems can grind your business to a screeching halt, devouring resources and devastating your quality of service.

Why hunt and peck, trying to recreate the problem and arguing about who's to blame?

AppSight breaks through the wall between the place problems are found and the place they're solved, so your team can pinpoint root causes faster than ever.

We're talking user blunders, configuration problems, performance issues, all the way down to code errors —

All without taking your application offline.

With AppSight,
it's problem solved.



JAIN/SLEE

Opening the telecommunications world for Java

by Sven Haiges

Another suitable title for this article would be “EJB for Communications.” The JAIN APIs still play a minor role on Sun’s Java Web site, but the JAIN initiative is getting stronger. The JAIN technologies (Java APIs for Integrated Networks) have the potential to radically change the existing service architecture for communications service providers.

Following the link “Other Java Technologies/JAIN APIs” from Sun’s Java Web site, you’ll probably find your way to a completely unknown world: Java technologies for telecommunication products and services. This article introduces you to the basics of JAIN and then explores JAIN/SLEE in greater detail. By the way, SLEE means Service Logic Execution environment and is abbreviated to JAIN/SLEE or JSLEE.

Motivation

The service architectures of many telecommunications service providers have become a lot more complex over the past few years. This is because many providers merged, or new technologies like GPRS or W-CDMA were or will be introduced. These architectures can be described as highly vertically integrated and heterogeneous. They miss common interfaces, based on standards. Instead, most architectures are quite proprietary. This makes it hard to introduce new services because the complexity of the overall architecture is augmented with each new feature that is added to the communications network. Services like the Multimedia Messaging Service (MMS) were introduced last year and again new protocols and

new network elements had to be integrated with the existing networks. Managing the communications network as a whole is getting harder and harder.

The JAIN initiative tries to break these vertical structures and replace them with horizontal ones. Proprietary interfaces between the network components are replaced with standards-based interfaces. The



JAIN APIs are specified using the Java Community Process, which allows everybody to contribute to the specifications. These companies include IBM, Motorola, NTT, and Vodafone.

JAIN and Other Java Technologies

As JAIN is a new Java technology for most developers, it’s interesting to explore how it relates to other existing Java technologies such as the Java 2 Micro Edition (J2ME), the Standard Edition (J2SE), or the Enterprise Edition (J2EE).

Java technologies can be divided into server-side and client-side technologies (see Figure 1). J2SE and J2ME

target the client side. While J2ME targets mobile and resource-constrained devices, J2SE targets standard desktop systems.

On the server side, J2EE technologies target enterprise systems and JAIN technologies target communications systems. Based on the JAIN and J2EE specifications, communications services running in a SLEE may communicate directly with existing enterprise systems using RMI or any other protocol that the J2EE server exposes. In contrast, if an enterprise service needs to trigger some functionality exposed by a telecommunications service, it needs to communicate with a SLEE’s resource adapter. The resource adapter then maps the requests into SLEE events and routes these events to the services.

Applications running on a mobile device need not necessarily use J2ME to access functionality provided by a server. For example, for MMS, the MMS User Agent on the client device is typically not implemented in Java.

Let’s compare this to a desktop system. A client Web browser can also be implemented in any programming language. It will be compatible with the services offered on the server, as long as the protocols such as HTTP are supported.

Unlike J2SE, J2ME does not provide the functionality to connect directly to EJBs. This is because RMI is not supported on resource-constrained mobile devices. To exchange information with external systems, mobile devices supporting J2ME may use HTTP connections.

The JAIN APIs

There are currently 36 entries for JAIN



Sven Haiges is studying computer science and economics at the University of Applied Sciences in Furtwangen, Germany. He obtained his MBA from SFU in Vancouver, Canada, and is currently writing his thesis for his German studies about JAIN/SLEE in Munich, Germany. Besides mobile Java technologies and communications, he specialized in Web Frameworks such as Struts and JavaServer Faces. He is the author of two books about these frameworks and has published articles in *JDJ* and the *German Java Magazin*.
sven.haiges@flavor.de

ULC

Rich Thin Clients for J2EE

UltraLightClient offers a server-side API to Swing, providing rich GUIs for J2EE applications.

- **Server-side programming model:**
develop scalable web applications for thousands of users as simply as stand-alone Swing applications.
- **Superior security:**
no application code is executed on the client, nothing is stored in a browser cache.
- **Application deployment on server:**
a lean Java presentation engine on the client serves any number of applications.
- **Pure Java library:**
use your favorite IDE and get add-on tools for visual editing, client/server simulation, and load/performance testing.



technologies on the JCP Web site. This number is more than we can cover in this short article, but we will categorize those specifications in general and then focus on the JAIN/SLEE specification.

JAIN technology enables the integration of Internet and Intelligent Network (IN) protocols, which are referred to as Integrated Networks. JAIN APIs can be divided into Java application interfaces and Java application containers. Table 1 provides an overview of some related JAIN technologies.

The Java application interfaces for communications map the telecommunication protocols for the Java programming language. In contrast, the Java application containers for communications provide a standard execution environment for telecommunication services. These services typically use

which may be distributed throughout the entire network. Because of this, Intelligent Networks allow a fast and efficient development of new services.

A protocol that's currently popular is SIP because it enables VoIP gateways; right now we can see that VoIP is beginning to change the telecommunications landscape radically. To talk "SIP," JAIN offers you the JAIN SIP 1.1 APIs as part of the Java APIs for Communications. SIP is an IETF protocol for IP-based communication. Using SIP, you can build your own SIP services like a SIP gateway, which is needed to create and manage the connections.

The Java Application Containers for Communications include only two specifications: SIP servlets and JAIN/SLEE. SIP servlets provide sup-

Why We Need a New Technology...

Communications systems are typically event-driven, asynchronous systems. In contrast, enterprise systems typically use direct method invocations. An existing enterprise architecture is defined by the Enterprise JavaBeans specification. The SLEE specification specifies an asynchronous, event-driven communications architecture that targets communications systems specifically. Table 2 provides a high-level overview of the different requirements of enterprise and communications systems.

As shown in Table 2, there are substantial differences between communications and enterprise systems. The EJB specification meets the requirements of enterprise systems. The SLEE now meets the requirements of today's communication systems.

“The service architectures of many telecommunications service providers have become a lot more complex over the past few years”

the Java application interfaces for communications via resource adapters.

Intelligent Networks are used in telecommunications systems. For example, call management for voice is done via the SS7 protocol. An intelligent network is also a service-independent network. The intelligence is not in the switch that connects to calling partners but in an external computer node,

port for SIP based on the well-known Java servlets standard. The JAIN/SLEE technology is a container for telecommunications services and provides a common runtime environment for those services. The specification went final in March 2004 and can be downloaded from the JCP home page together with the reference implementation and the TCK.

Although existing J2EE containers also support asynchronous event processing (JMS), these containers were not designed for it. A SLEE, on the other hand, was specifically designed for high-frequency telecommunications systems and is completely asynchronous. Thus, a SLEE fulfills the requirements of communications systems far better than any implementation on top of an EJB container.

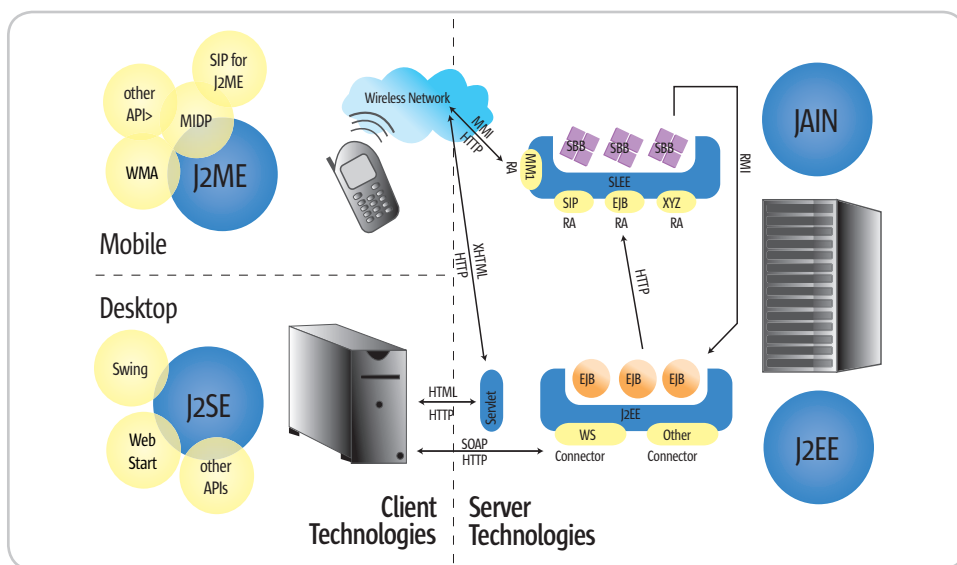


Figure 1 Client and server technologies

Service Logic Execution Environment

The Service Logic Execution Environment (SLEE) API Specification defines an application framework for the development of portable telecommunication services. It was specified under the Java Community Process (JCP) as Java Specification Request (JSR) 22. The final approval ballot accepted the specification on February 17, 2004, and it went final. The specification is led by David Ferry from Open Cloud and Swee Lim from Sun Microsystems. In addition to the companies that the specification leads belong to, the expert group for this JSR includes companies like Siemens AG, IBM, Motorola, and NTT Corporation.



Give your data direction:

- XML/Database/EDI data integration
- Drag-and-drop data processing / mapping
- Automatic code generation
- Instant output preview



DATA TRANSIT	
1	XML Line
2	Server Line
3	OOP Line

Next Stop, mapforce 2004!



Find your way to **mapforce™ 2004**, the premier XML/DB/EDI data integration tool from Altova! **mapforce 2004** is an award winning, visual data mapping utility that auto-generates custom mapping code in multiple output languages, including XSLT, Java, C++ and C#. With the power to map any combination of XML, Database and EDI into XML and/or Databases, **mapforce 2004** is the definitive tool for data integration and information leverage. **Download mapforce™ 2004 today: www.altova.com**

ALTOVA®

www.altova.com

All other trademarks are the property of their respective owners.

The SLEE reference implementation was built by Open Cloud, New Zealand. Open Cloud also created the reference implementation for JSLEE and sells an implementation of the SLEE that's not built on top of the EJB architecture (as the reference implementation is). Their product is called Open Cloud Rhino.

The four basic elements of the SLEE are resource adapters, events, activity contexts, and the runtime environment, which hosts the SBB Objects. Figure 2 shows the relationship between these elements.

The resource adapters are responsible for communicating with the external network protocols. They can send and receive events. Upon receipt of an event generated in the external network, they submit this event to the activity context as event objects. The SBB located within the SLEE runtime environment has interfaces to the activity contexts. The activity contexts are used to deliver these events to the SBBs. As resource adapters can communicate bidirectionally, they can also emit events to

the native protocol stack. Such events could be fired by SBB objects running within the SLEE that responds to incoming requests.

The activity context is a logical entity within the SLEE that receives and routes the events to the SBB components. The routing is exactly performed by the event router, which is part of the SLEE. Events may be duplicated and routed to several SBB components.

Resource Adapters

Resource adapters communicate with external systems to the SLEE, e.g., network devices, protocol stacks, directories, or databases. According to the SLEE architecture, a resource adapter is a vendor-specific implementation of a resource adapter type. An instance of a resource adapter within the SLEE is called a resource adapter entity.

The resource adapter type declares all event types that may be fired and all activities that the adapter introduces. When a resource adapter passes an event to the SLEE, it must

provide the event object, the event type, and an activity. The specification does not state how this information is passed to the SLEE. This API is up to the implementor of the specification. Because of this lack of clarity, a new JSR that should clearly define a Resource Adapter Framework was introduced in March 2004.

Events

Events objects carry information from one entity within the SLEE to another. Only SBB entities can both consume and produce events, while other entities such as resource adapters, the SLEE itself, and SLEE facilities can only produce events.

Each event is represented by an event object (subclass of java.lang.Object) and an event type. The event type determines how the SLEE will route the event, e.g., which SBB objects will receive the event to their event handling methods.

For each event that an SBB fires, the developer needs to specify an abstract fire event method. This method is implemented by the SLEE. SBB entities receive events from attached activity contexts. In case of an initial event, the SLEE first creates an SBB object and then routes the event to the SBB.

Activity Context and Co

The activity-related classes consist of the two logical entities, activity and activity context, and their Java object representations, activity objects and activity context interface object.

An activity represents a related stream of events. The Java representation of this logical entity is the activity object, which is created by either resource adapter entities or SLEE facilities. An example of an activity object is the JccCall Activity object. It's part of the Java Call Control APIs and represents a phone call.

An activity context represents the underlying activity within the SLEE and also holds shareable attributes that SBB entities want to share. The SBB objects can access the activity contexts through the activity context interface object.

An SBB can either use the generic activity context interface or extend this interface and define additional

Java Application Interfaces	Java Application Containers
for communications	
JAIN Session Initiation Protocol 1.1	JAIN Service Logic Execution Environment
JAIN Java Call Control 1.1	JAIN SIP Servlets
JAIN Presence	
JAIN Instant Messaging	

Table 1 Some important JAIN APIs

SOURCE: JAIN API SPECIFICATIONS, 2003

	Communications	Enterprise
Invocations	Mostly asynchronous, events generated through protocol triggers	Mostly synchronous; databases, EAI systems
Event Granularity	Fine-grained events and high frequency	Course-grained and low-frequency
Components	Lightweight components containing little business logic; rapid creation and deletion	Heavyweight data access objects that can have long persistent lifetimes
Data Sources	Multiple data sources (e.g., protocol-triggered events)	Database servers and back-end systems
Transactions	Lightweight transactions	Database transactions
Computation	Compute-intensive	Database access-intensive
Availability	3 to 5 9s	2 to 3 9s
Real Time	Soft real time	-
Deployment Distribution	Distributed throughout network	Centralized in small number of data centers
Nodes	1-4 CPUs	2-32 CPUs
Clusters	2-16 nodes	2-4 nodes

Table 2 Requirement comparison of event-driven and enterprise systems

SOURCE: [FERRY, PAGE, LIM, O'DOHERTY, 2003]

Achieve higher quality in less time, with fewer resources.



Parasoft® Automated Error Prevention (AEP) Products: Specifically designed for under-staffed, over-committed development organizations like yours.

Parasoft AEP Products ensure that comprehensive error prevention practices are applied consistently and uniformly across your entire team.

By automating compliance to a unified set of testing and coding standards, Parasoft AEP Products enable every team member to follow the same best practices and the same error prevention techniques – including coding standards analysis, unit testing, code review and regression testing.

Automated testing and standards compliance for any team configuration.

Parasoft AEP Products automate error prevention for Java, C/C++, Web Services, the Microsoft® .NET Framework, Embedded Development, Web Development, Database Development and more.

For details go to: www.Parasoft.com/AchieveQuality

Call: 888-305-0041 x3307 Email: AchieveQuality@Parasoft.com



Founded in 1987, Parasoft's clients include IBM, HP, DaimlerChrysler and over 10,000 companies worldwide.

Availability

Parasoft AEP Products are available on Linux, Solaris, and Windows NT/2000/XP.

Contact Parasoft for details and pricing information.

Parasoft Corporation
101 E. Huntington Dr., 2nd Flr.,
Monrovia, CA 91016

“The JAIN initiative opens the telecommunications world for the Java programming language”

attributes that it wants to share with other objects.

The activity objects are typically generated by network events. The resource adapters listen to these events and create the appropriate activity objects. These objects are placed in the activity context of the SLEE. The SLEE is now responsible for the delivery of the generated events to the SBB objects. Vice versa, an SBB object can access the activity context interface to get access to the current activity object, e.g., a JccCall Activity Object. It can then fire events on this object, which will be delivered back to the resource adapters and to the underlying network.

To get access to an activity object, the SBB developer typically accesses the activity context interface, which is automatically available within every event handling method of the SBB abstract class.

Runtime Environment and SBB Abstract Class

According to the specification, the SLEE runtime environment must make some APIs available to the SBB components at runtime. Only a minimal runtime environment is specified, leaving it up to the implementor to provide additional functionality.

Currently, the SLEE must make

only the following APIs available to instances of SBB components: Java 2 Platform, Standard Edition, v1.3 APIs; JNDI 1.2 Standard Extensions; JSXP 1.0; and JDBC 2.0 Standard Extension (support for row sets only). As in the Enterprise JavaBeans Specification, the SLEE must not allow components to access the local file system. Yet, an SBB may open a socket connection or queue a print job.

The SBB abstract class is part of the SBB component (which also includes the local interfaces and the SBB deployment descriptor) and contains the event-processing logic, which has to be added by the SBB developer. The SBB developer implements an SBB abstract class for every SBB component. The runtime environment is responsible for creating the pooled SBB instances from these abstract classes. This process can be compared to the creation of an EJB component. As with EJB, the runtime environment will be responsible for implementing certain abstract methods and for creating instances that process incoming events.

Each SBB abstract class implements the `javax.slee.Sbb` interface and must be defined public and abstract. The concrete methods contain the application logic of the component,

while the abstract methods deal with firing events, container-managed persistence (CMP), child relationship management, profile CMP methods, and accessing the specific activity context. The abstract methods are implemented by the SLEE. It uses introspection and data from the deployment descriptors to create this specific code.

Conclusion

The JAIN initiative certainly opens the telecommunications world for the Java programming language. Presently, though, some APIs for telecommunications APIs are still missing (e.g., MM1 for multimedia messaging), which means that the developer has to implement these APIs. Further, the interfaces between these protocols and the SLEE (resource adapters) is not yet specified in the current JSLEE specification, which makes it almost impossible for third-party resource adapter vendors to enter the market (and actually create one).

Yet the JAIN Days held recently in Munich at Sun Microsystems in Germany allow a positive conclusion. An active community is evolving, especially around SIP, and the mentioned problems with the Resource Adapter Framework were transformed into a new JSR that should fix this problem. International telecommunications companies such as Vodafone and NTT DoCoMo are watching carefully and some have already been successful with some JAIN/SLEE-based service implementations. ☛

Resources

- “JAIN and Java in Communications.” Sun Microsystems. March 2004.
- JAIN API Specifications, 2003 and 2004: http://java.sun.com/products/jain/api_specs.html
- Ferry, D.; Page, D.; Lim, S.; and O’Doherty, Phelim. 2003. “JAIN SLEE Tutorial.” Sun Microsystems: <http://jainslee.org/downloads/jainslee-tutorial-04.pdf>

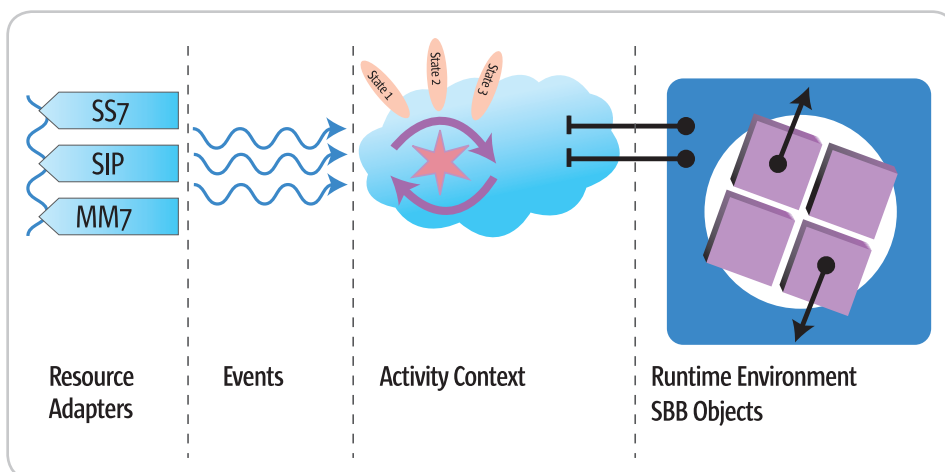


Figure 2 The four basic elements of the SLEE

We'd like to think that not all perfect matches are made in heaven.

Crystal Reports 10

Which edition of Crystal Reports® is right for you?

	Report Author/IT Editions		Bundled Developer Editions		Full Developer Editions	
	Standard	Professional	.NET Edition ²	Java ³ Edition ³	Developer	Advanced
Report Creation						
Visual report designer for rapid data access and formatting	●	●	● ¹	● ¹	●	●
Customizable templates for faster, more consistent formatting	●	●			●	●
Repository for reuse of common report objects across multiple reports ⁴		●			●	●
Data Access						
PC -based and Microsoft® ODBC /OLE DB for MS Access and SQL Server	●	●	●	●	●	●
Enterprise database servers (ODBC, native)		●	● ¹	● ¹	●	●
Custom, user-defined data through JavaBeans™				●	●	●
Custom, user-defined data through ADO and .NET			●		●	●
Report Integration						
Report viewing APIs (.NET and COM SDKs)			●		●	●
Report viewing APIs (Java SDK)				●	●	●
Extensive report viewer options (DHTML, ActiveX, Java Plug-in, and more)					●	●
APIs for run-time report creation and modification						●
Report Parts for embedding report objects in wireless and portal apps	●	●			●	●
Report Deployment						
Crystal Reports components for report viewing, printing, and exporting:						
a) Java reporting component				●	●	●
b) .NET reporting component			●		●	●
c) COM reporting component					●	●
Full featured report exporting		●			●	●
Report server (Crystal Enterprise Embedded deployment license)						●

¹ Limited functionality. ² Bundled with Microsoft® Visual Studio® .NET and Boland® C#Builder™. ³ Bundled with BEA WebLogic Workshop™ and Boland® JBuilder®. ⁴ This feature is available on the Crystal Enterprise CD, included in the Crystal Reports 10 package.

The Business Objects logo is a registered trademark of Business Objects SA. Copyright © 2004 Business Objects SA. All rights reserved.



Perfect matches can be made here too. In order to quickly determine which Crystal Reports® best suits your project requirements, we've provided this basic feature chart. Crystal Reports® 10 simplifies the process of accessing, formatting, and tightly integrating data into Windows and web applications via an enhanced designer, flexible data connectivity options, and rich Java™, .NET, and COM SDKs.

To learn more about Crystal Reports 10, compare over 150 different features across versions, or to access technical resources like the Developer Zone and evaluation downloads, visit: www.businessobjects.com/dev/p7. To ask more specific report project related questions, contact an account manager directly at 1-888-333-6007.



Beyond Entity Objects

Modeling concepts with objects



by Bill Kohl

Thus in object-oriented programming we have the view that computation is simulation.

—Timothy Budd

We have all read that objects are software representations of real-world entities and that one of the first design tasks is identifying these entities in our problem domains. These entities then become classes of our applications. However, the object-oriented paradigm allows us to model not only entity objects, but any abstract concept for which behavior can be identified. This article explores how to logically model abstract concepts with objects and why this will deliver higher quality software that is more maintainable and extendable.

Entity and Conceptual Objects

According to Merriam-Webster, an *entity* is “something that has separate and distinct existence.” Grady Booch in his book *Object-Oriented Design* refers to “a tangible and/or visible thing.” Common entity objects in business applications are Customer, Employee, PurchaseOrder, City, State, etc. Common practice is that db tables are represented as objects in the application (if not directly in the application, then in the translation layer between the application and the db). All of these are entity objects. They have a real tangible existence in the problem domain. *Conceptual* refers to objects that represent ideas that have no entity counterpart in the problem domain.

They may be identified in the vocabulary of the problem domain or they may appear only in the solution domain as design contrivances for solving the problems at hand. Validation, process, and builder are examples of conceptual objects. Our desire is to separate the conceptual ideas from the entities of the problem domain by objectifying them.

Creating Conceptual Objects

The first object-oriented programming (OOP) language

was Simula. As its name implies, Simula was used to develop simulations of real-world systems. The idea was to create software entities called objects that represent the entities of a real system, program each object to simulate the actions of its real-world counterpart, and then, through objects sending messages, let these objects interact to simulate the real system. Simulating entities with objects requires identifying the properties of the entity and its interactions with other entities of the system. If we were simulating a bowling ball, for example, it would have weight, shape, diameter, and color properties. It has interaction with the bowler who bowls it, the lane upon which it rolls, and the pins that it strikes. It would therefore respond to the messages *bowl* (velocity, spin) sent by the bowler, and *roll* (lane, time delta) and *strike* (angle) sent by the lane.

Simulating concepts works in a similar way. First we identify the properties of the concept, then we identify its responses to logical interactions with other objects.

A list, for example, has its elements as properties. Its logical interactions would be to answer queries about its size or one of its elements, or add or remove an element. For an integer, its property is its binary representation inside the computer. Its logical interactions are asking for its string representation or to compare itself to another integer object.

Concepts Become Objects

Now we can say what the conditions are for simulating abstract concepts. When I talked about making lists and integers into objects, I asked the following: What are the properties and what are the logical requests (behaviors) we could expect of a list or an integer? These are the same questions we ask of any concept we wish to objectify. But these requirements, though simple, are too restrictive; they can be refined further. Before doing so let's review encapsulation.

Encapsulation hides data and implementation; it does not hide the interface. Where does that leave us with the required conditions for simulating concepts? It



Bill Kohl works as a software architect for a large petroleum industry corporation. He has worked in software for more than 30 years, the last 15 of those in the OO world. His roles have included OO instructor and mentor; Smalltalk, C++, and Java developer; and for the last six years, software architect. He has extensive experience in developing object models for enterprise applications.

william.kohl@
crosscountryenergy.com



A business process may generally be defined as anything a business does that includes discrete events occurring over time”

means that the only requirement for simulation is that we can identify logical requests (interface) that the concept object should respond to. There do not need to be properties for the object. Most objects do have properties, and we should still be asking the question: “What are the logical properties of the concept we are simulating?” Now, however, the answer may be none. An example would be a NetworkBuilder, an object that implements the rules for creating a network. The rules are pure logic and the NetworkBuilder object contains no properties. All the data contained in the network is passed in as arguments; the NetworkBuilder constructs the network and passes back a Network object.

Cohesion, Coupling, and Encapsulating Change

You’ve heard the dictum, “maximize cohesion, minimize coupling” (*Structured Design* by Ed Yourdon and Larry Constantine). These programming rules, originally formulated for structured programming, are equally valid for OO programming. A lesser known rule is to “encapsulate change” (*Thinking in Java* by Bruce Eckel). That is to say, those areas of an application that are most likely to change should be contained in as few objects as possible. Conceptual objects can help us accomplish both of these goals. I always begin by asking the question, “What is most likely to change in the application?” If it’s a business application the answer will be business rules and business processes. Based upon the rule to “encapsulate change,” I would like to encapsulate business rules in one set of classes and business processes in another set. This would encapsulate change and increase cohesion since rules would have their own set of classes as would process. How do we do this? Rules and process are both abstract concepts. How can we create objects from them?

We could first observe that many business rules are applied by validating data entered into a system. The data may create new entities – a new employee, for example; or couple existing data – add Employee A to Dept. X. In either case, business rules are applied through validation, so validation becomes the concept we wish to objectify. Note that the examples I used above form two distinct cases for validation. In the case of validating a new employee, the data that creates the new employee is cohesive, that is it’s all contained in one object, an Employee object. Since the business rules associated with creating this new object involve only that object, they may be included with the behavior of the Employee class and we say that an Employee is self-validating. In the case where Employee A is added to Dept. X, the business rules involve at least two objects and probably more. (A HumanResources object may also be involved.) If the validation logic resides in the objects being validated, they are coupled by the rule logic and, furthermore, the rule itself is fragmented, parts of its logic lying in two or more classes. The concept of validation can be used to create a better situation here. The idea is to create a Validation class (this will probably become a hierarchy in the application) that encapsulates a business rule that would ordinarily be fragmented across two or more objects. This provides cohesion for the rule and decouples the objects involved in the rule.

Process Objects

One interesting result of being able to simulate abstract concepts with objects is process objects (as in a business process).

A business process may generally be defined as anything a business does that includes discrete events occurring over time. There is an initial event of the process and all succeeding events are determined by the finishing condition of previous events. An event is a discrete step in the process that can occur without interruption. When we consider business processes, it’s common practice to have objects collaborate to carry out this process. For complex processes this leads to a network coupling between objects of the process and highly fragmented process logic. Each object of the process must know about the next object of the process, thereby coupling the objects. If the process logic changes, its multiple locations must be tracked down. In this situation the concept of process leads to process objects that contain the process logic, thus increasing the cohesion of the system. Although the level of coupling has remained the same, coupling through the process object is more desirable, allowing us to alter the process, adding or deleting tasks without altering the entity objects of the process.

What are validation and process objects? How do we logically arrive at their structure? We begin by asking the same questions we asked in simulating lists and integers: What are the properties? What are the behaviors? I will put off the properties question for now and answer

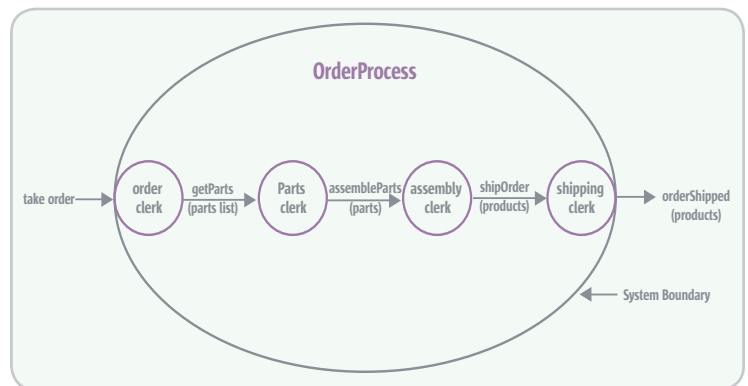


Figure 1 Order process with four objects

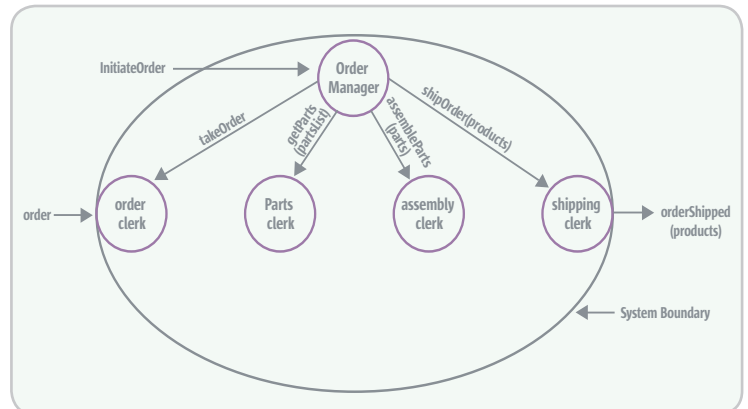


Figure 2 A process object

the behaviors question with an example, that of an order-fulfillment process. In Figure 1 we see an order process that includes four objects. Each of the objects participates in a part of the process and knows which object follows it in the process and what message to send to that object to continue the process. These objects are coupled by the process logic and the process logic is fragmented across these objects.

Figure 2 shows the same process with a ProcessObject coordinating the process. The objects involved in the process still carry out the tasks that represent their part of the process. However, they don't have any knowledge of the overall process.

They don't know what objects follow them or that they are even a part of a process. The OrderProcess object knows the entire process and is responsible for carrying it out. Process logic is encapsulated in the OrderProcess object and the four objects taking part in the process are decoupled with respect to the process. The OrderProcess object represents the entire process, but has delegated individual tasks to other objects.

Figure 3 shows how the PartsClerk object may participate in more than one process. It shows both the Order-Process and a Financial object that is determining the value of a company's assets. These functionalities are orthogonal, having only the collaboration with the PartsClerk object in common. By encapsulating the logic of each process in one

object, the PartsClerk doesn't need any knowledge of either process and contains logic related only to its own purpose.

The behavior of the OrderProcess object is the flow logic of the process. In our OrderProcess example, when each step of the process is completed, the OrderProcess object initiates the following step, which constitutes the behavior of the object. What are the properties of the OrderProcess object? Does it have any data properties? It has a name as a possible data property, but that's contained in its class name. Since it is conceptual and doesn't represent any real-world entity, it has no physical properties. Are there any data properties required for its behavior? The answer is probably yes, but if we take the simplest case it's no. In the simplest case the process could be "hard coded" with the logic (hardly ever a good idea) and the OrderProcess would not require any properties.

Instead of hard coding the process, properties of the process object could include a collection of process nodes, each of which knows how to initiate and finalize its step of the process. We would also include a DataDictionary to contain the data required for completion of the process. In our example, there would be four process nodes, one for each step (see Figure 4). When the process is initiated, it sends a "begin(this)" message with itself as an argument to the TakeOrder process node. The process node in turn notifies the OrderClerk object to "takeOrder(this)". When the order has been taken, the

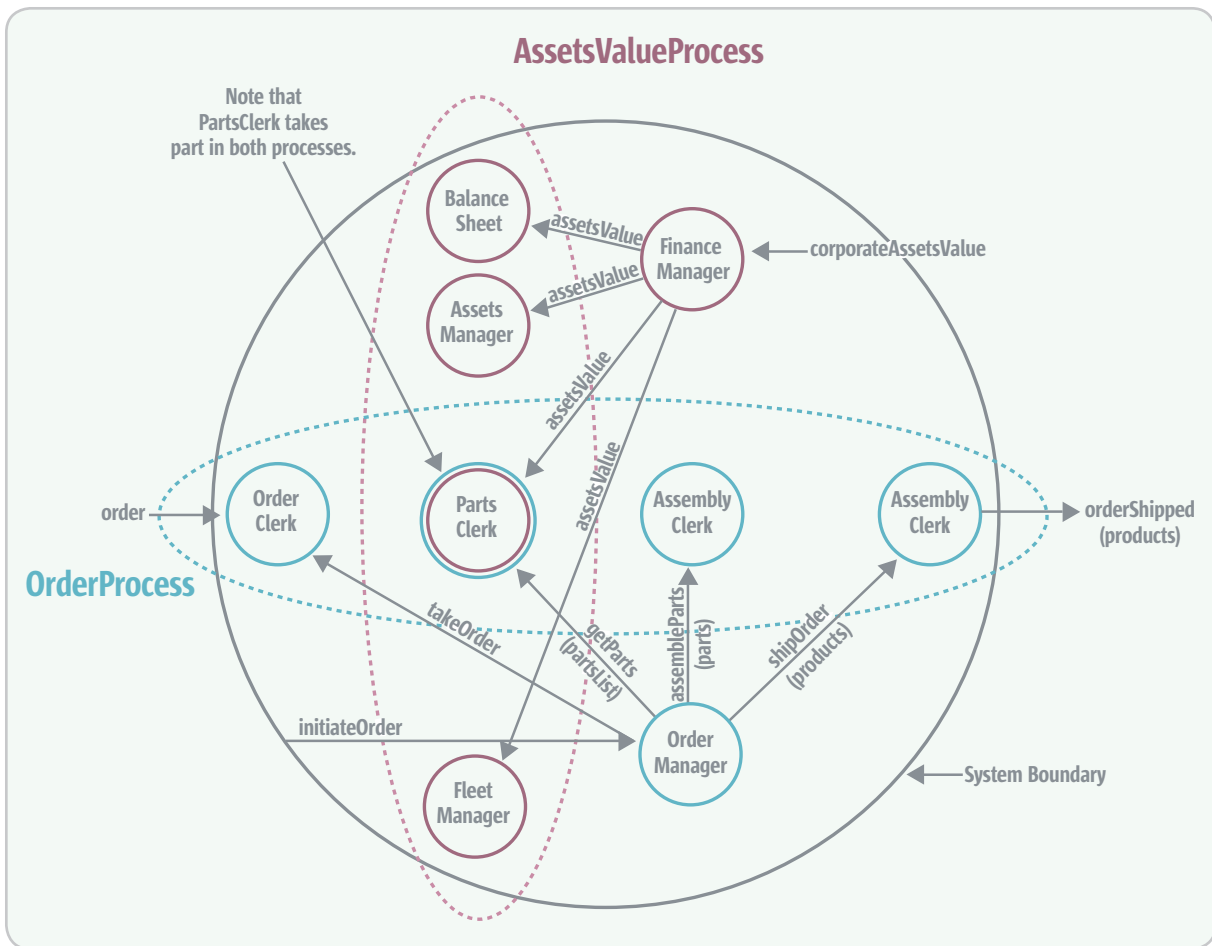


Figure 3 PartsClerk object participating in more than one process



TANGOSOL™



A Cure for Downtime.

Eliminate the single points of failure, add fault tolerance and enable transparent failover for your J2EE applications.

TANGOSOL COHERENCE™ has no single points of failure, ensuring high availability and reliability for your mission-critical J2EE applications. Application state, HTTP sessions and data caches are never at risk. Applications using Coherence keep it up without interruption, even when servers go down.

Try before you buy.

Download a free evaluation license at

www.tangosol.com

Coherence™ is pure Java software that reliably and cost-effectively scales J2EE applications across any number of servers in a cluster. Coherence provides reliable coordinated access to in-memory data that enables safe data caching and user session management for applications in a clustered environment.

Sales +1.617.623.5782 or sales@tangosol.com
Support support@tangosol.com or <http://www.tangosol.net>
Pricing US\$1,995 to US\$4,995 per Coherence production CPU
Development licenses are offered free of charge.

OrderClerk notifies the TakeOrder node “orderComplete()”, which in turn notifies the ProcessOrder. The ProcessOrder then sends “begin(this)” to the next node, etc.

All ProcessNodes implement a ProcessNodeInterface so that adding a node to the process would simply require inserting into the proper place in the node sequence. One property of the ProcessOrder would be this collection of process nodes. In addition we may wish to place some constraints on the process. This would lead to properties such as beginTime, endTime, and maxElapsedTime. If the process elapsed time exceeds the maxElapsedTime, the OrderProcess object would send notification of this condition to the appropriate party. These self-timing ideas would also apply to the process nodes. (You might already be imagining some additional benefits of process objects. The OrderProcess object could collect information about itself. This could be used to determine how efficient the process is, allowing us to improve the process over time.)

For complex processes, the process object may contain as a property a network that represents the process as task nodes and the state conditions dictating the next task of the process. This naturally leads to the idea of a subsystem of objects that form a framework for creating process subsystems.

What about the validation object? A validation object is simpler than a process object because it represents a single event. Its behavior is the application of the business rules via its methods. Properties may or may not be present. In the example above, a HumanResources object is required for the validation along with the employee and department objects. The validation class containing this rule may maintain a reference to a HumanResource object as a property. In general, validation objects will not have properties.

All of the data necessary to carry out a validation will be passed in as arguments. For example, an employee has certain information. The department may require that employees have certain skills before they can work there. The HR department would apply the process of validation by introducing the employee to the department where the department could interrogate the employee to determine if there is a match. Each department may ask different questions or they may all ask the same question and expect different answers; that information behavior is delegated to the department. The supplied object must conform to the Employee interface and, in doing so, decouples itself from the department. In turn, the department is decoupled from the implementations of Employee so that new types can be introduced with minimal effects on the code.

Conclusion

Developing object-oriented applications is more than modeling entities of the problem domain as objects. To achieve the goal of creating maintainable and flexible apps, we need to create objects from the abstract concepts of the domain and solution spaces that complement the entity objects of the domain. We find them in the vocabulary of domain experts, in

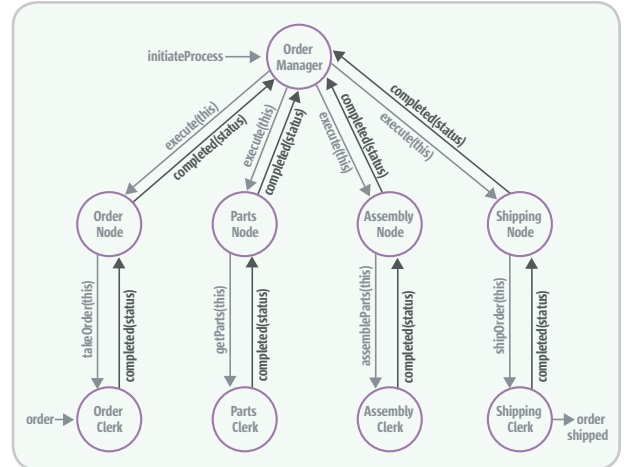


Figure 4 Four process nodes

requirements, in use-cases, in applying patterns, and in our own experience and knowledge shared by design experts. Our goal is cohesive objects whose purpose in the application is well defined. We strive for smaller objects that specialize. To get there we create objects such as Validation and Process. By objectifying these abstract concepts we increase the cohesion and reduce the undesirable types of coupling from the application. I’ve attempted to show how objects are created from abstract concepts. This is the first step in being able to create object-oriented applications that are composed of systems of cohesive, purposeful objects whose interactions fulfill the requirements.

For other ideas of conceptual objects, let me recommend the Wirfs-Brock reference below. Her “roles” should be helpful to you in finding conceptual objects in your applications. ☪

References

- Nygaard, K. “Early History of Simula,” (1978). *History of Programming Languages*. Richard L. Wexelblat, ed. Academic Press, 1981.
- Budd, T. (1997). *An Introduction to Object-Oriented Programming*. Addison-Wesley.
- Wirfs-Brock, R. (2003). *Object Design*. Addison-Wesley.
- Yourdon, E., and Constantine, L. (1978). *Structured Design*. Prentice Hall/Yourdon Press.
- Booch, G. (1991). *Object-Oriented Design*. Benjamin/Cummings.
- Martin, R., published articles: www.objectmentor.com/resources/articleIndex
- Next Step Object-Oriented Programming and the Objective C Language, Addison-Wesley, 1993: www.toodarkpark.org/computers/objc
- Merriam-Webster Online: www.webster.com
- Eckel, B. (2002). *Thinking in Java, 3rd Edition*. Prentice Hall PTR: www.faqs.org/docs/think_java/TIJ3_t.htm

“Developing object-oriented applications is more than modeling entities of the problem domain as objects”

Javavavoom!

Introducing a high-performance database that's 100% Java.

Berkeley DB Java Edition

Download at www.sleepycat.com/bdbje

Finally there's a high-performance database that loves Java just as much as you do: Berkeley DB Java Edition (JE). Brought to you by the makers of the ubiquitous Berkeley DB, Berkeley DB JE has been written entirely in Java from the ground up and is tailor-made for today's demanding enterprise and service provider applications.



Berkeley DB JE has a unique architecture that's built for speed. The software executes in the JVM of your application, with no runtime data translation or mapping required. Plus Berkeley DB JE has been specifically designed to handle highly concurrent transactions, comfortably managing gigabytes of data. And because it's built in your language of choice, your organization enjoys shorter development cycles and accelerated time-to-market.

Experience the outstanding performance of Berkeley DB JE for yourself.

Download Berkeley DB JE today at www.sleepycat.com/bdbje. Register now, and you'll also receive a 15% discount on a commercial license purchased before November 30, 2004.



Connecting the Java World to Grid-Enabled Databases

by Kuassi Mensah

Consolidate IT resources and optimize usage

Grid computing is not necessarily a new concept; however, its adoption within the enterprise has given birth to a new concept called enterprise grid computing, which is being embraced by the entire IT industry. Enterprise grid computing aims to consolidate IT resources – including both infrastructure software and applications – and optimize their usage, cutting costs substantially along the way. Since Java and J2EE are widely used as enterprise software platforms, how do they align with this vision?

This article outlines a set of challenges that JDBC faces as the database connectivity layer within enterprise grid environments and illustrates how the Oracle Database 10g JDBC driver addresses these challenges. First, I'll introduce the concept of enterprise grid computing; then I'll examine how Java and J2EE operate in grid environments and identify their database connectivity requirements; and finally, I'll discuss the features of the Oracle Database 10g JDBC driver that address those requirements.



Enterprise Grid Computing

Commercial software vendors use an assortment of terms, such as utility computing, on-demand computing, and autonomic computing, to describe enterprise grid computing. Regardless of which term is used, products that support enterprise grid computing all have the same set of common functional denominators:

- Resource consolidation
- Resource virtualization
- Policy-based resource management
- Provisioning or allocation

The chief aim of enterprise grid computing, as noted above, is to help enterprises consolidate IT resources and optimize usage, reducing infra-

structure and labor costs. This is a slight but important divergence from the original concept of grid computing now classified as academic grid computing, in which massive arrays of computational power are constructed from a network of many small and widespread computers, and used to perform large calculations and operations broken in autonomous chunks that can't be achieved even on today's largest supercomputers. A good example of this is the SETI@home project.

The convergence of recent hardware and software advances has made resource virtualization possible and allowed the enterprise grid to be constructed. On the hardware side, these advances include networked storage devices (like storage grids) and low-cost, modular hardware components (such as blades); on the software side, they include improvements in networking, Web services, databases, application servers, and management frameworks.

Although no specific enterprise grid computing standards have been estab-

lished, there is a general move toward the concept of service-oriented architectures (SOA), which are based on and make extensive use of existing Web services standards and specifications. SOA makes it possible to construct architectures where client applications can simply register, discover, and use the services deployed over the grid. This move is spearheaded by academic and research proposals such as the Open Grid Service Architecture (Global Grid Forum) and middleware vendors through participation in Web services standards bodies. A new consortium, Enterprise Grid Alliance, has been formed with the goals of developing enterprise grid solutions and accelerating the deployment of grid computing in enterprises.

Figure 1 illustrates a typical enterprise grid platform. While a complete enterprise grid computing discussion is beyond the scope of this article, I hope I've given enough information to encourage you to learn more about it.

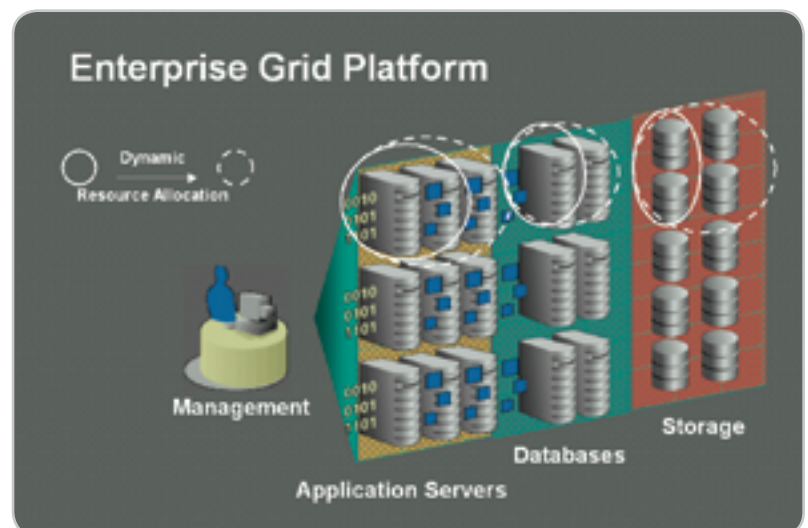


Figure 1 Enterprise grid

Kuassi Mensah is group product manager within the Java/J2EE and Web services platform group at Oracle. Mensah holds a MS in computer sciences from the Programming Institute of the University of Paris VI. He has published several articles in Java, J2EE, and Web services-focused publications and is a frequent speaker at industry events.

kuassi.mensah@oracle.com

Java and J2EE Containers in the Enterprise Grid

Very few enterprises run their business with just stand-alone Java or J2EE servers. Typical business applications employ more complex architectures consisting of load balancers, HTTP listeners, Web caching servers, J2EE containers, directory servers, resource managers and monitors, and management services.

In recognition of this, vendors, such as Oracle with its Oracle Application Server 10g, are actively supplementing their infrastructure with provisioning, monitoring, registering, discovering, and manageability mechanisms. This allows them to be used in the more complex system architectures required by modern SOA-based business applications.

Similarly, the J2EE world, realizing that it needs to broaden its support to interoperate with other applications and environments in a loosely coupled manner, is aligning with enterprise grid computing and SOA concepts. It's doing this by including core Web services standards (such as SOAP WSDL, UDDI,

and JAX-RPC) and numerous other Web services specifications including WS-I, WSIE, security, transaction, reliable messaging, events, peer-discovery, policy, orchestration, choreography, provisioning, and service-level agreement.

In the scientific and research worlds, the Globus Toolkit, from the Globus Alliance, is a reference implementation of the Open Grid Services Infrastructure (OGSI). OGSI lets Java components be exposed as OGSI services. OGSA-DAIS (Data Access and Integration) is another scientific and research proposal aimed at addressing data services requirements in grid environments; leading database vendors are participating in this initiative.

Enterprise grid computing creates an extremely dynamic environment where resources are automatically augmented or diminished based on business priorities captured in allocation policies. The current version of the JDBC specification (and implementations based on it) makes working in such environments impractical and reduces the ability of Java and J2EE ap-

plications to achieve all the benefits of enterprise grid computing. To be more explicit, let's look at some of the issues faced when working in an enterprise grid computing environment:

- Most JDBC applications today use connect strings that explicitly specify the hostname or IP address of the target database. This won't work when processing nodes are dynamically added to or retrenched from the cluster of database servers.
- Enterprise grid environments must scale well and serve thousands of concurrent clients. This makes efficient load-balancing and database connection pooling mandatory; traditional connection pooling mechanisms that cache only connections with the same identity are useless.
- With databases and other resources being dynamically managed and allocated for service, connection retention and management is critical. Mechanisms to resize or refresh stale connections or to search for and reclaim abandoned ones are required.

Oracle Grid

turns 64 PC servers into a giant mainframe

It's fast. . .
it's cheap. . .
and it never breaks

ORACLE®

oracle.com/grid
or call 1.800.633.0753

Copyright © 2004, Oracle. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

- Database clients, whether they're Java programs or J2EE containers, must be notified about changes in the database server configuration as quickly as possible. Application servers (and J2EE containers) that rely on unpredictable timeout mechanisms will not be able to provide a reliable, guaranteed service.

Java Database Connectivity Within the Enterprise Grid

Virtualizing the Database as a Service

As shown in Figure 1, clusters are pervasive at every level within the enterprise grid: disk and network storage servers, database servers, and application servers. Most database vendors now allow a single database to be concurrently managed by multiple servers or instances.

Although database clusters are easily understood, there are fundamental differences between the architectures supported by the various current implementations; for example, a shared nothing approach and a shared

disk approach provide different levels of capabilities. This article focuses on the Oracle Database Real Application Cluster (RAC) model in which any database server instance can address the entire database space, hosted on shared networked storage.

The typical JDBC approach to specifying how to connect to a database is to provide the details for the target database in the form of a tuple containing <host>:<port>:<sid>. This is set as the URL attribute of a data source definition or directly in the connect string when establishing the JDBC connection.

```
String connString="jdbc:oracle:thin:@prodHost:1521:ORCL";
```

In the Oracle Database RAC model, the physical server hosts and database instances that make up the cluster are represented as a single logical entity known as a service. A client application connects to the logical database service without any real knowledge of the actual host or database server instance that's

used in the cluster. By connecting to the service, as illustrated by the Enterprise Grid Model in Figure 2, and not to an actual physical server, clients can be insulated from changes made to the physical elements of the cluster as well as from node failures.

To use a logical database entity in Java applications, JDBC connection strings must be able to accept service-based connect strings instead of physical host details. In the Oracle Database RAC model, services are represented as "/<service-name>".

```
String connString="jdbc:oracle:thin:@/service_name";
```

The Connection Cache Manager

In the enterprise grid world, where applications are dynamically provisioned to run on different servers and even on instances of Java Virtual Machines (JVMs), it's imperative that the application's connection caches can be managed to permit the most efficient use of available resources. One approach to this is a new component called the Connection Cache Manager, which offers a centralized way to manage one or more connection caches. A single instance of Connection Cache Manager per JVM manages all of the connection caches used by applications running on that JVM.

The Connection Cache Manager plays two major roles: it manages cache and binds a connection to the data source.

Managing and Maintaining Cache

The Connection Cache Manager is responsible for creating the cache, maintaining its state, and terminating it. It's aware of the existence of each connection cache, managed independently. A rich set of APIs is provided to perform the Connection Cache Manager tasks.

The Connection Cache Manager supports the coexistence of more than one cache. Each cache is identified by a unique name and is then tightly bound to a data source. Each cache is either created transparently when getConnection() requests are made on a cache-enabled data source, or is created explicitly in the middle tier via the Connection Cache Manager API. Once a cache is created, it may either be explicitly removed

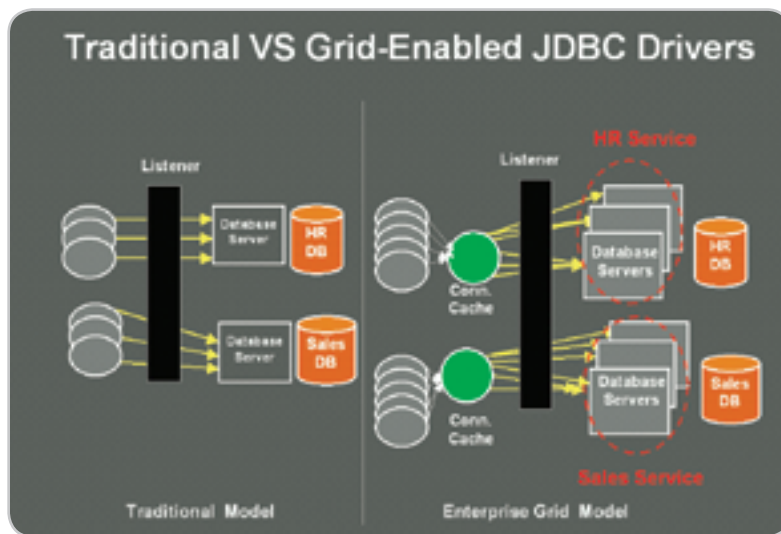


Figure 2 JDBC connections in a grid environment

Features	Oracle9i/R2 JDBC Traditional Connection Cache	Oracle Database 10g JDBC Implicit Connection Cache
Transparent cache access	No	Yes
Refresh stale connections	No	Yes
Attributes-based connection retrieval	No	Yes
Reclaim and reuse abandoned connections	No	Yes
Cache heterogeneous pairs of user/password	No	Yes
Centralized cache management	No	Yes

Table 1 Comparing Oracle JDBC drivers

via the Connection Cache Manager, or when the data source is closed.

Caches can be monitored using the Cache Manager APIs, enabling access to information such as the number of connections checked out and the number of connections available in the cache.

Binding a Connection Cache to the Data Source

As illustrated in Figure 3, the Connection Cache Manager makes sure a connection cache is associated with its data source object every time a cache is created, removed, or reinitialized. This ensures an efficient way to access the connection cache and retrieve connections from it every time the `getConnection()` method is invoked on a data source.

Fast Application Notification (FaN)

As I alluded to earlier, the dynamic nature of an enterprise grid environment results in nondeterministic system changes that may have flow-on effects on applications that are executing.

To allow application clients in an enterprise grid environment to function in the most efficient manner, a mechanism must be provided that performs the following tasks:

- Monitor changes in database configuration and notify clients as fast as possible.
- Balance connection requests across all active instances.
- Fail-over established connections to surviving instances

Oracle Database 10g JDBC offers these capabilities by combining the Connection Cache Manager, Oracle Database RAC, and notification events through a notification service to quickly inform affected components of changes occurring in the grid.

Let's look at how these pieces work together. When a new instance is added to a database cluster, or when an instance is retrenched from a cluster (instance stopped or node dies), Oracle Database RAC generates an event that indicates what happened. The Oracle Notification Service (ONS) detects and

distributes this event to components that have registered as interested in such events.

In Oracle Database 10g JDBC, Fast Application Notification is important: the connection caches are alerted when database instances are affected and can take preventive courses of action.

The mechanism is enabled on a cache-enabled data source by setting the data source property `FastConnectionFailoverEnabled` to true (see Listing 1).

With Fast Application Notification, event detection and notification is nearly instantaneous, occurring in seconds rather than minutes with traditional timeout mechanisms.

Adding Database Instances: Load Balancing

Adding new database instances to an Oracle Database RAC cluster generates UP events. These automatically trigger the balancing of all allocated connections over all active RAC instances without waiting for application connection retries/requests.

Oracle Database 10g

\$149 Per User

First class database . . . economy price

ORACLE®

oracle.com/standardedition
or call 1.800.633.0753

Terms, restrictions, and limitations apply. Standard Edition One is available with Named User Plus licensing at \$149 per user with a minimum of five users or \$4995 per processor. Licensing of Oracle Standard Edition One is permitted only on servers that have a maximum capacity of 2 CPUs per server. 17 minute install is based upon testing on a system with 1x866MHz Intel CPU, 512 Mb RAM running Red Hat Linux 2.1. Actual install times will vary and are dependent on system configurations. For more information, visit oracle.com/standardedition

Copyright © 2004, Oracle. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

Let's consider a basic example in which we have:

- A cluster with two nodes in a database service, one instance per node
- A cache size of 150 total connections, resulting in 75 connections per instance

Adding a new node to the service will trigger an UP event, which is detected and propagated via the FaN service. This will cause the connection cache to automatically rebalance the existing 150 connections in use in the cache over the three instances, resulting in 50 connections being established per database instance. This process involves removing some connections (to existing instances) and creating new ones (to the new instance).

Retrenching Database Instances (or Node Failure): High-Availability

Retrenching database instances from an Oracle Database RAC cluster generates DOWN events, which automatically trigger a detect-and-fix mechanism in the connection caches. This mechanism quickly removes connections belonging to the failed instance, preventing invalid connections from being handed out on connection request.

Continuing with the previous example:

- We now have a three-node database service with 50 connections per instance in the caches, for a total of 150 connections.

If a node in the database service fails or is removed by the resource provisioning system, then a DOWN event is created and propagated via the FaN service. The 50 connections belonging to the failed instance/node will be quickly removed from the cache, ensuring that the cache is clean and consistent. A clean cache guarantees that connection requests will be routed only to surviving

instances. Clients with a stale connection must retry connection requests, unless a container intersperses such calls and is able to take corrective action.

The Fast Connection Fail-over mechanism transparently ensures reliable, highly available Java database connections in RAC and grid environments.

In-Flight Transactions

If an application is midtransaction when an instance fails, it will be thrown an appropriate SQL exception and the transaction will be rolled back. It's the responsibility of the application or the container to retry the connection request and reestablish session state.

Simplifying JDBC Connection Caching

As summarized in Table 1, the new Oracle Database 10g JDBC Implicit Connection Cache has been designed to overcome existing JDBC Connection Caching limitations as listed in Table 1, by providing:

- Transparent access to the cache
- Support for multiple identities
- The ability to retrieve connections based on user-defined attributes and weights
- The ability to refresh or recycle stale connections from the cache

To take advantage of these capabilities, simply customize your environment by explicitly setting properties on the connection cache properties or connection.

Transparent Access to the Cache

By default, the `getConnection()` method in the standard `OracleDataSource` API creates a new database session and a physical database connection, thus incurring performance and scalability penalties. With Implicit Connection Caching, once the `DataSource` property `ConnectionCachingEnabled` has been set to true, the `getConnection()` method will service all connection requests from the connection cache.

```
ods.setConnectionCachingEnabled(True);
ods.setConnectionCacheName("MyCache"); //
optional
ods.setConnectionCacheProperties(cp); //
optional
ctx.bind("MyDS", ods);
ods = (OracleDataSource)
// lookup DataSource
ctx.lookup("MyDS");
```

A JDBC 10g Connection cache can be created either implicitly by the first invocation of the `getConnection()` method or explicitly by using the `ConnectionCacheManager` API. The `ConnectionCacheManager` is especially useful for developers working with J2EE containers and ERP frameworks since it shields the infrastructure from the complexity of managing connection cache.

```
// This call would create a "MyCache" cache
and a
// connection from "MyDS" data source will
be created and returned
conn = ods.getConnection();
```

```
// This call would create a "MyCache" cache
and a
// connection to "MyDS", authenticated by
"Scott" will be created
conn = ods.getConnection("SCOTT", "TIGER");
```

Subsequent `getConnection()` invocations will either create a new connection (if the cache was not previously initialized) or retrieve an existing connection from the cache. Once the connection is retrieved, you can proceed with statement creation.

```
// Create a Statement
Statement stmt = conn.createStatement ();
...
// Close the Statement
stmt.close();
stmt = null;
```

The cache can be populated in one of two ways: by preinitializing it using the `CacheManager` APIs or, incrementally, upon the release of connection(s) back to the cache. When returning a connection to the cache, applications can save current connection attributes settings for future use (see attribute-based retrieval, below).

```
// return this connection to the cache
conn.close();
conn = null;
```

Caching Multiple Identities

While a database does not impose any specific restrictions on the connection authentication, a traditional cache might impose a limitation on the connections it can manage, requiring that they all use the same username/password combination, for example.

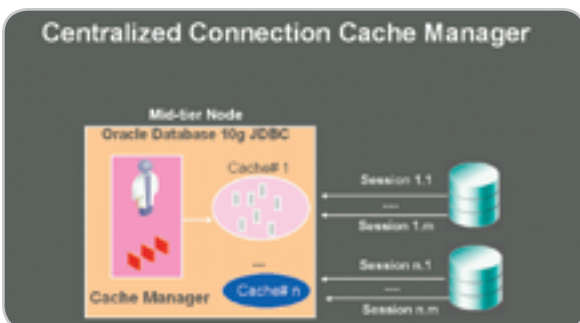


Figure 3 JDBC Connection Cache Manager

The Implicit Connection Cache can handle any combination of user-authenticated connections. For example, a joe.johnson connection can coexist very well with a sue.miller connection in the same connection cache.

Connection Retrieval Based on User-Defined Attributes

One of the valuable new features in the Implicit Connection Cache is connection striping.

Connection striping, or labeling, consists of applying user-defined attributes to a connection and making their state persist when the connection is returned to the cache. This speeds up future connection retrieval since cached connections don't have to reinitialize a block of state every time. These attributes can later be used to retrieve the same connection from the cache, as follows.

Example 1: Retrieving a connection based on the NLS_LANG attribute:

```
// get a connection from the cache with
NLS_LANG attribute
```

```
java.util.Properties connAttr = null;
connAttr.setProperty("NLS_LANG", "ISO-
LATIN-1");
conn = ds.getConnection(connAttr);
```

Example 2: Retrieving a connection based on the isolation-level attribute:

```
java.util.Properties connAttr = null;
connAttr.setProperty("TRANSACTION_
ISOLATION", "SERIALIZABLE");
```

```
// retrieve a connection that matches
Transaction Isolation
conn = ds.getConnection(connAttr);
...
// this call will preserve attr settings
for this connection
conn.close(connAttr);
```

Example 3: Retrieving a connection based on the connection tag and connection attribute:

```
java.util.Properties connAttr = null;
connAttr.setProperty("CONNECTION_TAG",
"JOE'S_CONNECTION");
```

```
// retrieve connection that matches Joe's
connection
conn = ds.getConnection(connAttr);
```

```
// apply attributes to the connection
conn.close(connAttr);
```

```
// This will retrieve Joe's connection
conn = ds.getConnection(connAttr);
```

Applying Connection Attributes to a Cached Connection

A connection attribute can be applied to a connection in the cache in two ways.

One approach is to call the applyConnectionAttributes(java.util.properties connAttr) API on the connection object. This simply sets the supplied attributes on the connection object. It's possible to apply attributes incrementally using this API, letting users apply connection attributes over multiple calls. For example, NLS_LANG may be applied by calling this API from module A. The next call from module B can then apply the TXN_ISOLATION attribute, and so on.

A second approach is to call the close(java.util.properties connAttr)

Oracle Application Server

More Features at Half the Price

	Web Services	Enterprise Portal	Workflow	Integration	Grid Ready	Wireless	Clustered Web Cache	Advanced Java Cache	Integrated Identity Management	Business Intelligence	Open J2EE Framework	BPEL Run Time
ORACLE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BEA	✓	✓	✓	✓	—	—	—	—	—	—	—	—

ORACLE®

oracle.com/more
or call 1.800.633.0759

Oracle Application Server 10g is half the price of BEA WebLogic Platform 8.1.

Comparison is based on Oracle Application Server 10g Enterprise Edition with Oracle BPEL Process Manager Option and Oracle Developer Suite 10g vs. BEA WebLogic Platform 8.1. Pricing reflects usage for 1 CPU and 1 Developer.

Copyright © 2004, Oracle. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

API on the connection object. This API closes the logical connection and then applies the supplied connection attributes on the underlying PooledConnection (physical connection). The attributes set via this close() API override any attributes set using the applyConnectionAttributes() API.

The following example shows a call to the close(connectionAttributes) API on the connection object that lets the cache apply the matched connectionAttributes back on the pooled connection before returning it to the cache. This ensures that when a subsequent connection request with the same connection attributes is made, the cache will find a match.

```
// Sample connection request and close
java.util.properties connAttr = null;
connAttr.setProperty("NLSLANG", "ISO-LATIN-1");
// request connection based on attributes
conn = ds.getConnection(connAttr);

// apply attributes to connection
conn.close(connAttr);
```

Connection Retrieval Based on Attributes and Weights

Connections may be selectively retrieved from the connection

cache based on a combination of ConnectionAttributes and attribute weights.

Weights are assigned to each key in a ConnectionAttribute in a one-time operation that also changes cache properties. The cache property CacheAttributeWeights is one of the java.util.Properties that allows the setting of attribute weights. Each weight is an integer value that defines how expensive the key is in terms of resources.

Once the weights are specified on the cache, connection requests are made on the data source by calling getConnection(connectionAttributes). The connectionAttributes argument refers to keys and their associated values.

The connection retrieval from the cache involves searching for a connection that satisfies a combination of the following:

- A key/value match on a connection from the cache
- The maximum total weight of all the keys of the connectionAttributes that were matched on the connection

Consider the following example in which a cache is configured with CacheAttributeWeights (see Listing 2).

(Listings 2–3 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

Once the weights are set, a connection request could be made as in Listing 3.

The getConnection() request tries to retrieve a connection from the MyCache cache. In connection matching and cache retrieval, one of two things can happen:

- *An exact match is found.* As in the above example, an exact match is a connection that satisfies the same attribute values and all the keys defined by Keys (NLS_LANG, SecurityGroup, and Application).
- *An exact match is not found.* In this case, the closest match based on the attribute key/value and its associated weights is used (but only if the ClosestConnectionMatch property is set).

Once the connection is returned, the user can invoke the getUnMatchedConnectionAttributes() API on the connection object to return a set of attributes (java.util.Properties) that did not match the criteria. The unmatched attribute list can then be used by the caller (or application) to reinitialize these values before using the connection.

Conclusion

This article highlighted a new set of Java data access requirements that have emerged when executing in an enterprise grid environment while also outlining how Oracle Database 10g JDBC tackles the challenges that these requirements present.

Whether you're using JDBC directly in applications via a run-time container such as an EJB container or through an O/R mapping framework such as Toplink, the latest Oracle JDBC drivers offer reliable and highly available data sources in RAC and grid environments and significantly simplify a range of data source connectivity issues for Java developers.

Oracle has proposed these new features to the JSR 221 (JDBC 4.0 specification) expert group. It hopes to see JDBC Connection Pool management and high-availability support for data sources within enterprise grid environments become part of the JDBC 4.0 standard. ☺

Listing 1

```
// Example to show binding of OracleDataSource to JNDI
// with relevant cache properties set on the datasource.
import oracle.jdbc.pool.*; // import the pool package
Context ctx = new InitialContext(ht);
OracleDataSource ods = new OracleDataSource();

// Set Datasource properties
ods.setUser("Scott");
ods.setPassword("tiger");
ods.setConnectionCachingEnabled(True);
ods.setConnectionCacheName("MyCache");
ods.setConnectionCacheProperties(cp);
ods.setURL("jdbc:oracle:thin:@(DESCRIPTION= (LOAD_BALANCE=on) (CONNECT_
DATA=(SERVICE_NAME=service_name)))");

// Enable fast connection failover
ods.setFastConnectionFailoverEnabled(true);
ctx.bind("MyDS", ods);
...
ds = lookup("MyDS"); // lookup datasource from the cache

// implicitly create connection cache, that is set up for fast
// connection failover
conn = ds.getConnection();
...

// return connection to the cache
conn.close();
...

// close datasource and cleanup the cache
ods.close()
```

MILLIONS OF LINES OF CODE
THOUSANDS OF DEVELOPERS TRAINED
HUNDREDS OF ENTERPRISE CLIENTS
TENS OF TOP-RANKED INSTRUCTORS
ONE COMPANY TO CALL

**SO YOU THINK YOU'RE A GOOD DEVELOPER?
THAT'S COOL. BUT WHY STOP AT GOOD,
WHEN YOU CAN TAKE IT TO THE NEXT LEVEL?**

Our passion is leading-edge technical skills transfer. Students tell us we're so good at our job because our instructors are not only excellent educators, they're also real-world architects with relevant enterprise experience. They're practitioners, they're authors, and they personally develop the most engaging courseware. But, we're not your typical talking heads, we provide consulting and mentoring for top-shelf technical teams. Our approach is practical, insightful, and challenging. The difference? **Results.** *Call us today at 888-211-3421 to discuss your specific requirements, or just visit us online.*

www.inferdata.com/jdjmag





Calvin Austin

Core and Internals Editor

Three Gems from JavaOne

You may have heard the news that Sun has opened the doors for its employees to start blogging, including the most famous employee, the COO. Blogging obviously isn't new, and many companies have already granted individual users the opportunity to go ahead. However, the open floodgate – simply to fill in your details on a Web page, press a button, and start posting – is proving an interesting challenge.

One item that has created some debate is what happens to the useful information after it gets posted and how can we prevent it from being lost forever. It's only human to discard most of a day's events, what we read and what we heard. How do we remember what is important and then how can we apply that to something like a sea of RSS-distributed content?

With that thought in place, and with the JavaOne conference now becoming a distant memory for me, I wanted to spend a little time this month walking through some of the Tiger features announced at JavaOne that didn't have a dedicated session. Features that perhaps are long forgotten or were missed in the first place.

APT

The first feature is a new tool in the Java Developer Kit called the Annotation Processing Tool (APT). APT is used to manipulate annotation tags defined by JSR 175. One way to think about annotations is to compare them to some of the original metadata-like tags in J2SE such as the `@deprecated` Javadoc tag. When put inside a Java comment, the Javadoc tool and Javac tools understood the `@deprecated` tag and had rules for what to do with one when they came across it. In the deprecation example the Javac tool would generate a compiler warning and the Javadoc tool would generate some special HTML text.

The new annotation API uses the `@interface` keyword to define a new annotation and allows you to specify properties to represent that annotation. In the case of the `@deprecated` example, one property could be the release number that the deprecation first appeared in. Then every time you used the `@deprecated` annotation you could supply the release as one of the properties, for example, `@deprecated ("1.4") mymethod()`.

How does this reduce boilerplate code and make developers' lives easier? This is where the APT tool comes in. The tool and its associated API can parse Java source files looking for Java annotations. By registering annotation processors with the tool, an annotation processor, which is simply a class that has a `process` method, can generate custom rules for each annotation. The annotation processor could be a very simple Javadoc-like tool that generates text output displaying the properties of the annotation it is parsing. The more advanced annotation processor could generate new code and the API could provide access to a `Filer` class that can help generate that boilerplate code in a new class, which is then passed to the Javac compiler.

For most users the developer tools are already gearing up for annotation and processing support so that the only task left for the user is to add the annotation and let the tool take care of the rest.

Diagnostics

We have a great monitoring article in the *JDJ* pipeline that will explain how you can now get low-memory alerts using JMX or SNMP technology and delve into the monitoring and management frameworks. However, there are also several small tools in the reference implementation that can be used to help track down simple issues.

One such tool is called `jps`. `jps -l` will list all the J2SE 5.0 reference JVMs running on that machine, which is especially useful on older Linux releases. The JVM ID or process ID that is returned from `jps` can then be used by other diagnostic or debug tools. Another new tool is `jstack`, which can generate a stack trace on a running JVM. If you supply the `jstack -m` option, Java and native code will be listed in your stack trace. Look for other tools like `jmap` and `jinfo` to provide further diagnostic information.

You can even generate a stack trace from your own program without having to throw and catch an exception. This is achieved with the new stack trace API; the easiest way to use it is to call `Thread.getAllStackTraces()` and then simply read the output.

Java Plugin

The final feature I want to shine a spotlight on is one of the small improvements to the Java Plugin. When you load an applet, the default color for the background is dull gray. In 1.4.2 a tiny coffee cup icon was placed in the corner of that gray background. In 5.0 the first step in launching an applet is to display the coffee cup logo on a white background with a rotating clockwise swirl. It also displays a progress bar tracking the classes that are being loaded. It not only looks better but gives download feedback to end users.

If you have heard of these three features before, I hope this was a useful reminder. If you're new to them, look out for future coverage in *JDJ*. On that note, a big thank you to everyone who has submitted J2SE 5.0 article proposals. As a *JDJ* reader I'm really looking forward to seeing the finished articles come through. ☺

A co-editor of *JDJ* since June 2004, Calvin Austin is the J2SE 5.0 Specification Lead at Sun Microsystems.

He has been with Java Software since 1996 and is the Specification Lead for JSR-176, which defines the J2SE 5.0 ("Tiger") release contents.

calvin.austin@sys-con.com

Yeehaw?



Some reporting solutions are pretending to be something they're not.

When you look under the fancy packaging, there's only one Java reporting solution with years of proven, real-world experience and thousands of successful deployments worldwide – JReport.

JReport is a comprehensive, 100% J2EE reporting solution that is intuitive in nature and easy to integrate, eliminating the need for costly professional services and training. From departmental applications to enterprise-wide deployments, JReport scales to meet your reporting needs, ensuring uninterrupted access to critical business information, so you get the answers you need, when you need them.

Whether you need to deploy sophisticated reports via the Web or enable users to create ad hoc reports on demand, JReport delivers on the promise of self-serve reporting. And, with JReport's *one-click analysis*[™], any user can slice-and-dice, pivot, and drill-down on a report, from any Web browser.

Visit www.jinfonet.com/yeehaw or call **301-838-5560** today, and discover why, when it comes to Java reporting, JReport is the real McCoy.



Java GoF Creational Design Patterns

For cleaner development and easier maintenance

by Puneet Sangal

A design pattern is a solution to a recurring problem. Although using patterns this way is well known and has been around for a while, it was only when the GoF wrote their famous book, *Design Patterns*, on software design patterns, that patterns slowly but surely became an industry standard.

A design pattern is not just object-oriented design, but the communication between these objects. The GoF Creational patterns are a specific subset of these patterns that create objects for you, rather than creating them directly. Why would you want that? It provides cleaner development and easier maintenance for your applications. I'll cover five creation patterns here, and attempt to help you learn and implement them. It's important to not only understand their usage, but also know when to use these patterns. (A basic knowledge of interfaces and abstract classes is assumed as a prerequisite for this article.)

I'll provide at least one simple way to implement each pattern, and at least one scenario in which you might use it. Of course, a complete description is beyond the scope of this article. The other two categories for patterns, as defined by the GoF are structural and behavioral. We won't discuss these categories here, but instead will keep our focus on creational patterns.

The Factory Pattern

This pattern returns an instance of one of several possible classes based on the data passed. This allows us to introduce new classes without modifying the code substantially. An immediate advantage here is that the instantiation of the class occurs inside the Factory class, which provides more flexibility in code maintenance and development than having to create an object directly in the client of this class.

Example

An example of a Factory pattern is the home interface of an EJB that instantiates a bean based on the data passed to it. Here's another simple hypothetical example. We have a Web site where users are required to enter their phone numbers. To keep our example simple, the phone number can be entered in one of two formats, as shown in Table 1.

Our implementation of the phone classes would look like Listing 1. In this listing we have a base abstract class and two subclasses. Our base class is abstract because we want to enforce instantiation of a subclass only. It's also not necessary to define a constructor for our



base class. We have a PhoneFactory class that decides which of the subclasses would be returned based on the argument that is passed to the factory. The returned object from our factory is one of the subclasses, depending on the data passed. Now all a client class needs to do is call the getPhoneNumber() method in the factory class with the phone number as its argument.

The Abstract Factory Pattern

An abstract factory is a factory object that returns one of several related factories. There can't be any simpler definition. This means that instead of returning one subclass, it returns an abstract class that can in turn return a family of its subclasses. Thus both composition and inheritance play

equally important roles here. The benefit of using this pattern is that it isolates the concrete subclasses from the client. This pattern should be used when the system is required to be independent of how the components are organized.

Example

We'll expand our future example of phone numbers. If we internationalize our Web page, we can have different phone number types for different countries (see Table 2).

The implementation of the classes is shown in Listing 2. We have more or less a similar structure as before, only a little more complex. Here we have a base abstract class and two subclasses. The subclasses implement the abstract methods of the base class that are capable of returning a PhoneNumber object. We have an InternationalPhoneNumberFactory class that decides which of the subclasses would be returned based on the arguments that are passed to the factory. As before, all a client class needs to do is call the getInternationalPhoneNumber() method in the InternationalPhoneFactory class with the phone number as its argument. The instantiation details and which subclass gets instantiated are hidden from the user. Once we have an international phone number object, we can still have phone numbers with or without spaces for both of the countries. We would call the "with" or "without spaces" method using our previous PhoneFactory class, if required.

The Singleton Pattern

This pattern is used when a class can have only one instance in a system. Think of a Web application when one application-level class is used to track a number of clients. Multiple classes per session are unnecessary; just one for all the clients would suffice. This pattern



Puneet Sangal has been working with Java for more than six years, focusing on enterprise Web-based Java applications. He is a Sun Certified Programmer for platform 1.4.
psangal@nlg.com



A design pattern is not just object-oriented design, but the communication between these objects”

can be achieved in a number of ways. For example, we can create an exception that will be thrown by a class if it's instantiated more than once. Next we'll have a boolean static variable in our class, because a static variable can be shared among all instances of a class. In the constructor of this class, we'll set this variable the first time it's instantiated. Any other time, since it is already set, the constructor would throw an exception if this variable is already set. This is an excellent strategy. But take care to de-set this variable when the instance is destroyed (in the finalize method). Different JVMs can exhibit different behaviors; using this would ensure we can instantiate an object of this class again after it is destroyed.

If a generic class is needed that only provides static methods, that class can also be declared as final. Using this final-static combination would prohibit any instance creation and allow us to use this class as a whole. Note that this is different from the earlier approach, where we actually created a single object of the class.

Another popular approach to creating Singleton patterns involves having a private constructor and a static method in a class. The private constructor would ensure that an instance can be created only from within the method of the class, which is static. The static method would return an instance and set a boolean variable indicating instance creation. As before, we would de-set this variable in the finalize method as a good coding practice.

These approaches are simple and direct, so I won't provide any example code for them.

The Builder Pattern

The Builder pattern allows a client object to construct a complex object by specifying its type and content only. The way in which objects are assembled can be achieved using a Factory pattern. The factory class used here is called Director, and the actual classes derived are called Builders. This pattern is similar to the abstract factory pattern because both return a family of objects. The difference, however, is that the abstract factory returns a family of related objects while the Builder pattern constructs a complex object one step at a time depending on the data supplied.

Example

I'll attempt to describe a simpler example in which a Builder pattern can be used without providing a Java template for it, as that is beyond the scope of this article (constructing a Builder pattern, Java example would consume 150–200 lines of code). Consider a fast-food restaurant like Burger King where they have a special meal for kids. Irrespective of whether the order is chicken, a hamburger, or something else, the meal always consists of food and a toy. Here the client is the customer, the cashier is the director, and the restaurant crew is the builder. The builder knows how to build the meal, the director knows what to build, but they don't know how to perform each other's tasks. This is a layer of insulation the pattern provides us, that each can be varied without affecting the other.

The Prototype Pattern

This pattern is used when you want to copy an existing instance of a class,

instead of creating a new one. Why?

Because there might be so much data already in the object that it will take a considerable amount of time to get it again with a new instance. We can use the clone() method of the Object class to create a copy. Of course, the objects that can be cloned need to implement the Cloneable interface.

Imagine a list of value objects that can be obtained from a database that will contain a thousand objects. Each of these value objects would have a lot of information. This data is displayed on a particular Web page and our business need states that the information it contains must be isolated from other database transactions until the user logs out. This information should be displayed in part on different Web pages. This calls for creating a clone. One thing to pay attention to while cloning is that any operation performed on the copied data will also occur on the original data because references to data objects are copied, not the objects themselves. In some cases, this might be unacceptable. In this case, our class would also implement a Serializable interface.

Now we would just write our object out to a stream and reread it, making the two objects completely independent of each other. This again assumes that all the objects contained in this class are themselves serializable. Listing 3 provides the deepClone() method.

Summary

I've discussed five creation patterns here. If you are interested in learning more about patterns, I would recommend reading one of the many books available about design patterns. ☺

References

- Sangal, P.M. "Using Interfaces and Abstract Classes": <http://java.ittoolbox.com/documents/document.asp?i=3063>
- *Java Creation Patterns*: www.fluffycat.com/java/patterns.html#creationalpatterns
- Cooper, J. "Java Design Patterns": www.patterndepot.com/put/8/JavaPatterns.htm

Phone Number	Description
XXX XXX-XXXX	Phone number indented with spaces between area code and the number
XXXXXX-XXXX	Phone number with no spaces between the area code and the number

Table 1 Phone number formats

Names	Description
USA	Ten-digit phone numbers. Still can have both above formats in the factory pattern example.
Spain	Nine-digit phone numbers. Also can have both above formats in the factory pattern example.

Table 2 Phone number types

Listing 1

```
package factory;

public abstract class PhoneNumber
{
    protected String m_AreaCode;
    protected String m_Number;

    public String getAreaCode()
    {
        return m_AreaCode;
    }
    public String getNumber()
    {
        return m_Number;
    }
}
```

```
package factory;

public class PhoneNumberFactory
{
    public PhoneNumber getPhoneNumber(String s_PhoneNumber)
    {
        if (s_PhoneNumber.trim().indexOf(" ") > 0)
            return new PhoneNumberWithSpaces(s_PhoneNumber);
        else
            return new PhoneNumberWithoutSpaces(s_PhoneNumber);
    }
}
```

```
package factory;

public class PhoneNumberWithoutSpaces extends PhoneNumber
{
    public PhoneNumberWithoutSpaces(String s_PhoneNumber)
    {
        this.m_AreaCode = s_PhoneNumber.substring(0, 3);
        this.m_Number = s_PhoneNumber.substring(3, s_PhoneNumber.length());
    }
}
```

```
package factory;

public class PhoneNumberWithSpaces extends PhoneNumber
{
    public PhoneNumberWithSpaces(String s_PhoneNumber)
    {
        this.m_AreaCode = s_PhoneNumber.substring(0, 3);
        this.m_Number = s_PhoneNumber.substring(4, s_PhoneNumber.length());
    }
}
```

Listing 2

```
package abstractfactory;

import factory.PhoneNumber;

public abstract class InternationalPhoneNumber
{
    protected String m_InternationalPhoneNumber;

    public abstract PhoneNumber getPhoneNumberWithSpaceFormat();
    public abstract PhoneNumber getPhoneNumberWithoutSpaceFormat();
}
```

```
package abstractfactory;

public class InternationalPhoneNumberFactory
{
    public InternationalPhoneNumber getInternationalPhoneNumber(String s_Country, String s_InternationalPhoneNumber)
    {
        if (s_Country.equalsIgnoreCase("USA"))
            return new USAPhoneNumber(s_InternationalPhoneNumber);
        else
            return new SpanishPhoneNumber(s_InternationalPhoneNumber);
    }
}
```

```
package abstractfactory;

import factory.PhoneNumber;
import factory.PhoneNumberWithSpaces;
import factory.PhoneNumberWithoutSpaces;

public class SpanishPhoneNumber extends InternationalPhoneNumber
{
    public SpanishPhoneNumber(String s_SpanishPhoneNumber)
    {
        this.m_InternationalPhoneNumber = s_SpanishPhoneNumber;
    }

    public PhoneNumber getPhoneNumberWithSpaceFormat()
    {
        return new PhoneNumberWithSpaces(this.m_InternationalPhoneNumber);
    }

    public PhoneNumber getPhoneNumberWithoutSpaceFormat()
    {
        return new PhoneNumberWithoutSpaces(this.m_InternationalPhoneNumber);
    }
}
```

```
package abstractfactory;

import factory.PhoneNumber;
import factory.PhoneNumberWithSpaces;
import factory.PhoneNumberWithoutSpaces;

public class USAPhoneNumber extends InternationalPhoneNumber
{
    public USAPhoneNumber(String s_USAPhoneNumber)
    {
        this.m_InternationalPhoneNumber = s_USAPhoneNumber;
    }

    public PhoneNumber getPhoneNumberWithSpaceFormat()
    {
        return new PhoneNumberWithSpaces(this.m_InternationalPhoneNumber);
    }

    public PhoneNumber getPhoneNumberWithoutSpaceFormat()
    {
        return new PhoneNumberWithoutSpaces(this.m_InternationalPhoneNumber);
    }
}
```

Listing 3

```
public Object deepClone()
{
    try
    {
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);
        objectOutputStream.writeObject(this);

        ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(byteArrayOutputStream.toByteArray());
        ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
        return objectInputStream.readObject();
    }

    catch (Exception ex)
    {
        System.out.println("Error in deep cloning: " + ex.getMessage());
        return null;
    }
}
```

Style Report 6

Solid Building Blocks for Enterprise Reporting & Analytics



Challenges:

- Too costly to start off and expand
- Too many UI to train
- Too complex to manage and support

Solutions:

- One step at a time, at your own pace
- One zero-client user interface
- One web application

One Integrated, Open Architecture Solution



For more information and to download a free evaluation copy www.inetsoft.com/jdj

Writing Java Card Applications

by Vijay Phagura
and Anita Phagura

Writing applications for the smallest JVM

This article describes a Java Card and how to write applications that can be accessed by enterprise applications. We'll discuss the complete development and testing process for card applications. The sample application and the code listings are kept simple for readability and easier comprehension of the basic ideas.

Java Card

More uses are being found for smart cards since their introduction about a decade ago. As the name suggests, these cards are smarter than the usual magnetic strip cards due to a built-in chip – either a memory chip or a microprocessor. Microprocessor smart cards are becoming more popular as they're more secure and can process information. A Java Card is a smart card with a microprocessor and the smallest Java Virtual Machine (JVM) called the Java Card Virtual Machine (JCVM). Please refer to Sun's specification on Java Card Virtual Machine specs. It makes sense to put a JVM on smart cards due to the following advantages of Java:

- A standard programming language
- Inherent security
- Multiple programs and applet instances can coexist on the card due to Java's security
- Integration with mainstream Java IDEs
- Benefits of object-oriented programming
- Platform and vendor independence

Card Hardware and Software

The cards with processors usually have an 8-bit microprocessor (similar to 6805 or 8051), although the new cards are coming out with 16-bit processors. They usually contain three types of storage:

- **ROM:** For persistent, nonvolatile, and nonalterable data
- **EEPROM:** For persistent, nonvolatile, and alterable data
- **RAM:** For nonpersistent, volatile, and alterable temporary data

ROM is usually used for system software that's masked with some preissuance data. EEPROM is used by post-issuance applications and some part of it can also be used by the system software. RAM, of course, is used as a scratchpad for temporary storage.

Most Java Cards have a native operating system that interfaces with the JCVM. The Java Card Runtime Environment (JCRE) sits on top of the JCVM and interfaces with the applications (see Figure 1).

The Java Card system supports a small subset of Java APIs that includes the following packages:

- java.lang
- javacard.framework
- javacard.security
- javacardx.crypto

Package java.lang

The Java Card java.lang is a subset of the JSDK java.lang. This package provides the basic support for the language, which defines the root class Object and some basic classes for exceptions.

Package javacard.framework

This is an important package for the Java Card; it defines the Applet class and the APDU class, which provides the infrastructure for communication. There are other supporting classes like the PIN class for PIN access and validation support. The Java Card java.lang package does not support the System class so the javacard.framework.JCSystem provides that support.

Package javacard.security

This package provides cryptographic support for the Java Card platform. It's based on the JSDK java.security package.

Package javacardx.crypto

This package provides interfaces and classes that are subject to U.S. export regulations. It's up to the JCRE provider to provide implementations to most of the classes in this package. A smart

card may have a separate coprocessor to perform cryptographic operations.

Card Protocol

The card communicates with the host via a card reader in a half-duplex manner. The communication packet is called Application Protocol Data Unit (APDU). The command is sent by the host and the card responds to it. The command APDU is called as C-APDU and response APDU is referred to as R-APDU.

The headers for a C-APDU and R-APDU are shown in Figure 2.

The C-APDU header contains 4 bytes:

- **CLA:** Class of instruction
- **INS:** Instruction code
- **P1 and P2:** Parameters 1 and 2

The optional body section varies in length:

- **Lc:** Specifies the length of the data field
- **Data field:** Data sent to the card
- **Le:** Number of bytes expected from the card

The R-APDU response to the card is as follows:

- **Data field:** Optional and contains data sent by the card, with the length specified by Le.
- **SW1 and SW2:** These form the status word. If the value is 0x9000, the communication was successful, otherwise this would represent an exception code.

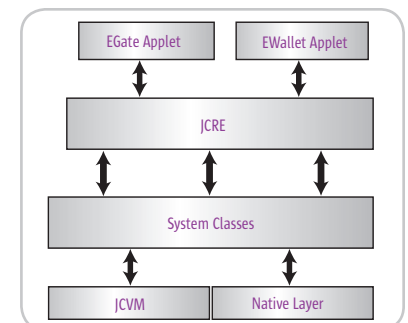


Figure 1 General architecture



Vijay Phagura,

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

vphagura@yahoo.com



Anita Phagura has more

than 12 years of experience in software development. She has designed and worked with many different Java and J2EE technologies and APIs. She also has experience with different GUI technologies. Anita has patents pending for some of her work in the software architecture and design arena.

anita_phagura@yahoo.com

Application Development Process

The JVM differs from the standard JVM as it's divided into two parts: the interpreter running on the card, and the converter running off-card. The class files are passed through the converter, which produces a converted applet (CAP) file, along with other files. The CAP file is loaded onto the card, which is then instantiated and executed using the interpreter. It's a two-step process to execute a Java Card application.

To download the Java Card Development Kit 2.1.1 (JCDK) visit the link in the Resources section. JCDK comes with a reasonably good documentation to get you started.

The Card Applet

The applications developed for cards are Java applets that extend from `javacard.framework.Applet`, which is one of the differences between a Java Card applet and a regular Java applet. The card applet has a private constructor that can only be instantiated by the JCRE. Also, it has different methods like `install`, `process`, etc., unlike the regular applet's `init`, `start`, `stop`, etc., methods. The `install` method instantiates the applet and

the other objects it needs when `select` is called. The `select` method prepares the applet to receive APDU commands from the host. The `process` method processes the APDU commands and prepares responses.

Each package and an applet are identified by an ID called AID. An applet can have an AID 5–11 bytes long. The first 5 bytes of an applet AID should be the same as its package AID.

Java Card Application

We'll now develop an application that will use the Java Card as an identification card for a temporary employee in an organization. The application will be loaded on the card along with his personal information. To retrieve the information that's not stored on the card for this individual (e.g., a start date and an end date of the person's contract, etc.), this application can access the database on the host system. Along with the access function, the card can be used within an organization for various other functions, e.g., an employee is allowed to purchase items from a company store on credit, and the amount due will be stored on this card.

The application designed here supports the following functions:

- Provides an ID for access at the entrance gates
- Allows or disallows entrance to the person by disabling the card. The card can be disabled by at least two conditions:
 - Contract expired, detected by off-card application
 - Not paying the dues on time, as will be discussed later
- Keeps track of amount due
- Keeps track of number of days left to pay the dues

There would be an attribute on the card that keeps track of the number of days the person has to pay his dues. Not honoring this would deactivate the card and forbid the person automatic entry (this condition may not be valid in a real-world application).

This application demonstrates that all the data for a person does not have to be on the card; a few things can be done by off-card applications on the host system. For instance, the card application does not keep track of when the contract is expiring; this can easily be done off-card. It's extremely impor-



Build your legacy.

*What lasting impact can you truly make? We asked this question as a company when we were chartered over three decades ago. Our answer? Our legacy will be seen in the eyes of families and their children, and in the communities we help shape. The people we serve are **energetic, goal-directed, and diverse** – just like the professionals who define our growing organization. We invite you to bring your own individual perspective to an industry leader whose clear mission is **to bring greater stability to the nation's mortgage markets, and to expand opportunities for homeownership and affordable rental housing.***

We have opportunities for professionals with **all levels of Java experience** to support our rapidly growing, state-of-the-art IT department.

To learn about our superior benefits, *including relocation assistance*, to apply online, and for full details on our opportunities, please visit our Web site at:

www.FreddieMac.com/Careers.

Freddie Mac is an equal opportunity employer who firmly supports and recognizes the value of diversity.

www.FreddieMac.com



tant to design the on-card applets carefully in order to respect the resource constraints.

The Code

Listing 1 provides the code for the card applet class EPersonApplet. It seems pretty long at first look, but it is simple. (Listings 1–9 can be downloaded from www.sys-con.com/java/sourcecode.cfm.) In brief, the code declares a lot of constants that are instructions for each command that the applet will process and these instructions represent the INS byte of the APDU word. Apart from the INS bytes there's an arbitrary CLA byte for this applet's APDU class and an AID. There are also six fields in this applet to hold the data.

The fields are:

- empId to store the Employee ID
- firstName to store the first name
- lastName to store the last name
- active to indicate whether the card is active
- amount to store the Amount due
- allow to store the number of days to pay the due amount

It's important to note that the fields can be of type byte, boolean, or short (of course, arrays are allowed). In the Java Card specs, support for integer is optional.

Next is the constructor. Note that it is declared as private for reasons described earlier. The actual data for each field is described in the comments in this constructor. The constructor takes a byte array as one of the parameters to initialize these fields. The initialization string is 28 bytes. For the sake of this example, each field has been assigned a length, for instance, the name fields can be 10 bytes long and so on. (*Note:* For testing this applet with the Java Developer Kit 2.1.1, the lengths may have to be reduced because of tool limitations, which takes fewer bytes

for initialization.) Some card platforms also send AID + 4 bytes along with the initialization parameters to the applet constructor. These extra 4 bytes are:

- Length of applet instance AID
- Length of application privileges
- Application privileges
- Length of application-specific parameters

The local variable short offset takes care of pointing to the initialization parameters, avoiding the AID + 4 bytes. The method Util.arrayCopyNonAtomic() is the card version of System.arrayCopy(). As the name suggests, it is array copy utility. Its atomic version is Util.arrayCopy(), which means the card supports transactions and guarantees that if a transaction fails, the original data is not lost; for instance, if the card is disconnected.

One important task for the constructor is to register the instance with the JCRE by calling the register() method of the super class.

The next method is the install(). As soon as the application is installed, the JCRE calls this method, which in turn creates an instance of this applet.

The process() method does all the command processing and in this applet it delegates by calling the appropriate method. This method is also called when the applet is selected by the JCRE. The rest of the code in this method is self-explanatory. It's important to note the exception handling: the class ISO7816 declares constants for the exceptions and the applets can call the throwIt() method on it to throw the exceptions.

The rest of the code implements the commands. The following describes how to send and get data to and from the card. The steps for the “set” commands or sending data to the card applet are:

- Get the APDU buffer reference by using apdu.getBuffer().
- Set the JCRE mode to receive data: let JCRE know the amount of data to receive and receive the data. This all can be done by calling apdu.setIncomingAndReceive() method or by separate methods.
- Check if the number of bytes received is correct.
- Copy the bytes to a local variable.
- Process the received data.

Similarly, the “get” commands or getting data from the card applet can be generalized in the following steps:

- Get the APDU buffer reference by using apdu.getBuffer().
- Set the JCRE mode to outgoing and let JCRE know the amount of data to send and send the data. This can all be done by calling apdu.setOutgoingAndSend().
- Convert data if necessary.
- Send the data using apdu.setOutgoingAndSend().

The getAllInfo() is a useful command for the application. To gather all the data from all the fields, prepare them into a byte array and send them. (As shown in Listing 6, this can be useful in creating client objects.)

Compile the applet using JSDK 1.3.x, with the Java Card JARs in the classpath, to get a class file.

Using the Java Card Development Kit

Now that we have the applet designed, it's time to test it. Testing can be done off-card for sanity and then loaded to a real card. For testing and simulation we can use the Java Card Development Kit (JCDK) 2.1.1.

The following are brief descriptions of the various tools provided by the JCDK.

Converter

The converter tool in the JCDK converts the class file to a CAP file, which is put on the card. The converter is run on the command line with many options, which are well documented in the JCDK. Another way to run it is to use an .opt file that has all the options defined. Listing 2 provides a sample opt file.

Please refer to the JCDK documentation for the description of all options used in the opt file.

The Converter produces other files too, apart from the CAP file. For now we're only interested in the CAP file. The following line shows how to start the Converter using an opt file.

```
converter -config egate.opt
```

JCWDE

This is the Java Card Workstation Development Environment tool that comes with the JCDK. It needs an .app file to run; a sample .app file is shown in Listing 3.

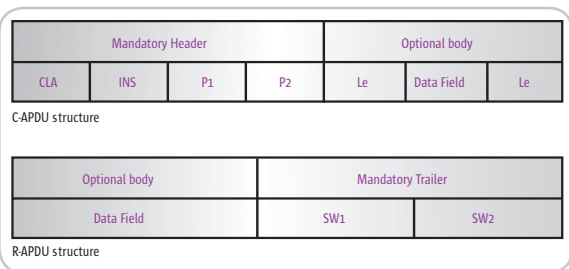


Figure 2 C-APDU and R-APDU header

This file lists the applets to be tested. There is an InstallerApplet that always gets loaded before any other applet. More info on this applet can be found in the JCDK documentation. The JCWDE tool can be started on a command line as follows:

```
JCWDE -p <port #> egate.app
```

After it has been successfully started it waits to listen on the specified port.

Apdutool

This tool executes the command script, the .scr file, and produces responses from the applet. A sample .scr file is shown in Listing 4. As seen in this listing, the .scr file is a script to run commands. The first few commands are routine commands as there is always an InstallerApplet on the JCDK to install other applets. After these commands the EPersonApplet is selected first before any commands can be issued on it. For more info on the format of the Installer commands, please refer to the JCDK documentation.

This .scr file can be fed to the apdutool, which in turn gives out the responses from the card for each command. Listing 5 shows a sample of these responses. The apdutool can be started as follows:

```
apdutool egate.scr > egate.out
```

The output can be sent to a file as shown above or, if the output filename is omitted, the output goes to the console.

As seen from this file, the bytes 0x9000 in the SW1 and SW2 indicate a successful response.

The JCDK is very useful for testing your card applet by simulating it on your workstation. It helps you develop and improve your application without burning up card resources.

These are the steps to test the applet using JCDK tools:

- Write an .opt file to run the Converter tool.
- Convert the class file into a CAP file using the JCDK Converter utility.
- Write an .app file, which describes

the applet to the JCWDE. This file lists the installer applet, its AID, and the applet under test and its AID.

- Start the JCWDE tool in a command window using the .app file.
- Write a .scr file, which contains all the APDU commands.
- Run the .scr file through the apdutool in another command window.

Real Card

Now that we have designed, developed, and unit tested our applet, we're ready to load it onto a real Java Card.

The one we used is called the CyberFlex e-gate 32K card from Axalto. This is one of the cards that supports Java Card 2.1.1. It has 32K of EEPROM space and the hardware specs are as follows:

ROM: 96K RAM: 4K EEPROM: 32K Enhanced 8-bit CPU with extended addressing modes Data retention: 10 years EEPROM endurance: 500,000 write/erase cycles Single power supply: 2.7V to 5.5V Icc supply current: MAX 50mA at 5MHz
--



FIND SOMETHING BETTER.



Technology is hot again. Is your career? **NOW** is the time to explore new opportunities.

Visit Dice.com to find a better job with better pay. Check your salary. Compare your skills. Search over 50,000 tech jobs from leading companies and choose to have new jobs emailed to you daily.

IT'S TIME for something better.
Visit Dice.com today.

DiceTM
Look to the tech leader first.TM

Axalto also has 64K cards. The 64K card and the 32K e-gate cards are both JavaCard 2.1.1 compliant. The main difference between the 32K and the 64K is the capacity. The 64K card currently does not have an e-gate feature. The final difference is that the 64K card does not have Codeshield (on-card byte code verifier), whereas the Cyberflex 32K e-gate does. There will be an e-gate version of the 64K card available later this year.

Axalto supports all of its cards with an SDK (Software Development Kit), which is very versatile and an all-in-one toolkit. We used the current release – SDK 4.5 version. It's not necessary to use the SDK; however, it includes a set of convenient tools for developing on-card and off-card applications. Specifically, the SDK provides a set of middleware interfaces that makes writing off-card applications very convenient.

For more information on these cards and the features of the SDK please visit the link at the end of this article. The user manual for the SDK is freely downloadable from their site.

Once you have decided on the card, use its software to load the applet onto it. Irrespective of the type of card or toolkit, it's important to note that loading an applet is a multistep process. One important step is to create an instance of an applet. Unlike the JSDK, where class instances are dynamically created, in the card it's not dynamic anymore. You have to create an instance manually before using the application through a toolkit. Before sending APDU commands to the applet on the card, the applet has to be selected. The selecting process can be done through a tool provided by the vendor or via an APDU command.

The applet can be tested using the card's toolkit. We used an Axalto SDK tool called APDU Manager to test the applet. You can send APDU commands from the tool to the card and also see the logs of responses sent by the card.

Off-Card Application

Now that we have tested our applet on an actual card, let's focus on the off-card application. The off-card application or host application will access the card applet or may be part of another larger application on the host.

These applications can be written using the Open Card Framework

(OCF), details of which can be found at the link given at the end of this article. The CyberFlex e-gate cards support this framework, but the functionality achieved is limited. To write this application we used Axalto's middleware libraries, which come with their SDK.

As mentioned earlier, this application can be part of a bigger enterprise application. This can be hooked as a proxy to the actual card and act as an API to the card applet. For this reason we named this application CardProxy. The CardProxy acts as a card listener to capture asynchronous events like card insertion, card removal, etc.

Listing 6 shows the complete code for the CardProxy class. The code has been kept simple, omitting thread safety and some error-handling scenarios.

Similar to the EPersonApplet, there are a few constants for the commands. Apart from that there are byte arrays that define the applet AID and the keys for the card. The keys are needed to establish a secure channel with the card before the applet is selected.

As seen in the code, the CardProxy class has to interface with JNI and hence loads the library in the static initializer. It also has to create an IOP object, static in this case, that represents the interoperable layer. The other important thing to note is the connect() method. In this method a connect() is called on the IOP object. Even if this call returns true, the system needs some time before the EstablishSecureChannel method is invoked. After a secure channel has been successfully established, the applet needs to be selected. After this is successfully accomplished, only then can we "connect" to the card. In case of an error, the error is acquired from the card by invoking the method getErrorMessage(). The connect() on the CardProxy is called from two places: the constructor and the CardInserted() callback. The constructor call to connect() is useful if the card is already inserted before starting the host application.

Most of this class contains getters and setters that use the method SendCardAPDU() on the SmartCard object. This method takes six parameters: command class code, instruction code, p1, p2, a zero length int array, and the length of the output data. The data is returned in a short array, each element of which contains a byte of

info. The 0th element contains the MSB and the *n*th element contains the LSB. The SendCardAPDU() is also used to set data onto the card. It's used exactly as described earlier except the zero length int array now contains the data to be set.

The command getAllInfo() is slightly different; it extracts all the info from the card and creates an Employee object. Listing 7 shows a partial code for Employee class. It's a simple class with accessors and mutators for the employee data.

The Client

The real host application to use all of the functionality of the CardProxy class is the Viewer class. Listing 8 shows a simple version of the Viewer that accesses all the commands of the card.

A client application can be as simple as the Viewer listed above or an RMI component with a JDBC object to access the database, and it communicates with an application server-deployed application. The basic idea for accessing a Java Card is the same. Listing 9 provides another version of the Viewer class, which is a stand-alone Swing applet.

Conclusion

Although Java Card technology is not that new, it's still unexplored territory for many. It is advancing fast and today there are more applications for it. We have tried to describe the full development process of a small project. We hope this article was motivating and provided some practical insight into Java Card application development.

Acknowledgments

We thank Axalto and, especially, David Teo and his team for providing excellent support and hardware, without which it would have been difficult to write this article. ☺

Resources

- *Java Card Development Kit 2.1.1 download*: http://java.sun.com/products/javacard/dev_kit.html
- *Axalto*: www.axalto.com
- *Axalto's Cyberflex Access Java Cards*: www.cyberflex.com
- *Axalto Middleware Guide, Cyberflex Programmer's Guide*: www.cyberflex.com/Support/Documentation/documentation.html
- *Open Card Framework*: www.open-card.org

Introducing

SOAPscope 3.0

Debug

Test

Tune

4 Ways to Know Your Web Services

Whether you are learning how a Web service works, or troubleshooting a tough problem, you need the help of a "smart" tool. SOAPscope lets you dig deeper, faster.

- 1. Try It** Solve problems by testing your Web service with different inputs without writing any code.
- 2. See It** View WSDL and SOAP to understand what's happening. Capture from any toolkit, and see just the right detail for the task at hand.
- 3. Diff It** Compare a problem message or WSDL with a similar, working one.
- 4. Check It** When the problem's not obvious, rigorous interactive analysis finds inconsistencies, errors, and interoperability problems.

Look What's in 3.0

- Integrates with Microsoft® Visual Studio® .NET and Eclipse
- Graph Message Statistics
- Interactive Message Analysis
- Interoperability Testing System
- SSL Support
- HTTP Authentication Support
- HTTP Compression Support
- Support for multi-byte encoding
- Best usability of any Web Services tool

The most comprehensive Web services diagnostic system available, and still **only \$99!**

Try It online at XMETHODS.net or download a trial at Mindreef.com

"SOAPscope 3.0 is easily the most addictive piece of software I've encountered since Halo. When does the multi-player version come out?"

Don Box
XML Messaging Architect,
Microsoft and co-inventor
of SOAP

© Copyright 2005, Mindreef, Inc. The names of companies and products mentioned herein may be the trademarks of their respective owners. This product uses Hypersonic SQL. This product includes software developed by the Politecnico di Torino and its contributors.

"The Web Services Diagnostic Experts" **Mindreef**

by Jim Menard

Building Applications with Berkeley DB Java Edition

High-performance database goes pure Java

Berkeley DB is a database with a long history. First released in 1991 as a replacement for various dbm implementations, it was soon included in BSD Unix releases. Requests for new features and commercial support led to the formation of Sleepycat Software in 1996. Using a dual license model, Berkeley DB became very popular for both open source and commercial applications. Recently, Sleepycat announced Berkeley DB Java Edition (JE), a 100% Java implementation that runs on any J2SE 1.4.2 or later JVM.

This article introduces JE's features and classes and uses a sample electronic voting application to illustrate its use.

Berkeley DB and JE

The original Berkeley DB is used in a wide variety of open source and commercial applications in telecom, Internet infrastructure, storage, security, financial services, and other industries. Sleepycat estimates that there are over 200 million copies deployed, but most people are not even aware they touched Berkeley DB. (See "Where's the Sleeping Cat?" available at the URL listed in References.)

Berkeley DB is a nonrelational database that stores key/value pairs. Databases can be configured to store multiple values under a single key. Secondary databases can be created that use different keys to access the same values. Searches and joins can be performed using a specific key or a range of keys.

The original Berkeley DB (not JE) is written in C, though there are bindings for other languages, including Java. JE brings to Java developers the features of Berkeley DB and adds better integration with the Java world through Java data type support while avoiding the performance penalty of JNI.

Like its C ancestor, JE is small. The database's footprint is less than 435K. It supports full ACID transactions (mean-

ing its characteristics include Atomicity, Consistency, Isolation, and Durability), performs record-level locking for high concurrency, and can handle huge amounts of data: hundreds of terabytes in a single table, with record sizes up to two gigabytes. Frequently used data is cached in memory. JE has better concurrency than the C Berkeley DB for some applications because, unlike DB, it has record-level locking that allows applications to continue accessing other parts of the database when a record is locked.

JE databases are stored as files on the file system. Backing up a database is accomplished by copying the files. To restore a database, move the copied files into the database directory and restart the application. When an application uses a database with transactions (that group database operations together in order to treat them atomically), checkpoints are periodically saved to the file system by a separate JE thread in order to make recovery faster. Applications without transactions may need to perform manual syncs that tell the database to flush data in memory to the file system, depending upon their requirements.

Technology Overview

Databases

JE databases are B+Trees whose records are key/value pairs. The keys and values are byte arrays. Databases are stored as files within a single directory. An in-memory cache stores as much as it can of the B+Tree structure and frequently accessed data.

The class DatabaseStats provides information about a database, such as the number of records in the database or the depth of the B+Tree.

Environments

A database environment coordinates one or more databases. It man-

Configuration

While creating or opening databases, JE allows you to specify many optional database configuration parameters, including but not limited to:

- Maximum database file size
- Whether transactions are allowed
- Transaction timeout length
- Cache size
- Whether duplications are allowed
- How frequently to perform checkpoints
- How to sort duplicate records

All of this is done through the Java API. You don't need a database administrator to create or maintain your databases.

Why Berkeley DB Java Edition?

Where should you use Berkeley DB Java Edition? Consider using JE if:

- Speed and size are issues. JE is fast and can handle huge amounts of data.
- It makes sense to embed the database into your application. Some applications are delivered to platforms that don't have a database installed or can't connect to a database server.
- You know what questions you will be asking of the data. Writing queries involves writing Java code and designing the database structure to allow access to the data based on keys other than the primary key.

JE may not be the solution if:

- The application needs to perform ad-hoc queries. For this you need to write Java code to perform searches; there is no other query language. Searches that use keys other than the primary key used to store the data require a secondary database, discussed later in this article.
- Data must be shared with non-Java applications. The original Berkeley DB data format is different than the JE format. Berkeley DB supports bindings to many languages, including Java.



Jim Menard is a senior technologist with over 20 years of experience in software development, design, and management. He has developed many open source projects including the Java GUI report writer DataVision (<http://datavision.sourceforge.net>). Jim likes shiny things (www.io.com/~jimm).

jimm@io.com

ages transactions, allowing them to work across databases. The environment provides the in-memory cache used by an application. It also facilitates administrative operations such as renaming or deleting databases.

Records and Binding

Each record in a database is an instance of DatabaseEntry, essentially a wrapper around a byte array. Since storing Java objects is a common operation, JE provides ways to translate between them and byte arrays. This process is called binding. JE provides classes for binding simple numeric and string objects.

There are two approaches to creating DatabaseEntry values from more complex Java objects: serialization or creating a custom class that translates the objects to and from byte arrays. When serializing objects to be stored in separate database records, each serialized instance carries with it duplicate information: the descriptions of the classes involved in the serialization. JE provides binding APIs that perform the serialization and store the duplicate information only once, in a separate database (see Figure 1).

A custom binding that maps objects to byte arrays must subclass TupleBinding, which implements the EntryBinding interface. This interface consists of two methods: objectToEntry and entryToObject.

Cursors and Iterators

To look up a single key, call myDatabase.get(), passing in the key as a DatabaseEntry and another DatabaseEntry to hold the returned data.

To iterate through the database or a subset of the database, both a cursor and iterators are available. They operate the same way, searching for a (possibly partial) key and iterating over the returned values. The difference is that iteration uses flavors of the Java collection classes called StoredMap, StoredSet, and StoredIterator. Unlike java.util.Iterators, StoredIterators must be closed explicitly. There are two ways to close a StoredIterator: call its close method or call the static method StoredIterator.close. The latter approach avoids casting when the iterator is stored in a variable of type Iterator.

The JE collection classes implement all of the collection methods except for size (for example, get, put, and

Database Log Files

Databases are stored as a series of log files. When data is written to a disk, it's always appended to an existing log file. This enables very fast writes since the disk head does not need to move. Log files are never overwritten or modified; those operations are slower than appends. When the log file in use by a database reaches a maximum size, a new log file is created. A background "cleaner" thread organizes the log files, moving active records from older log files to newer ones and deleting old log files that contain no active data.

containsValue) and a few extras. See the Javadocs for details.

Secondary Databases

To perform searches using keys that are different than those used to store the data, a secondary database must be created. The values in the secondary database are the same as those in the primary database. Multiple secondary databases may be created.

When a primary database is modified, JE updates all of its secondary databases. All writing takes place through the primary database. The one excep-



Went to find himself.

Left you 300K lines of Java linked to 225K lines of C++ using macros to look like Pascal, and a Linux box you can't boot.

www.scitools.com

Tools that help you understand and maintain impossibly large bodies of source code.

tion to this rule is that records may be deleted from a secondary database. The records in the primary database are also deleted, as are all corresponding records in all other secondary databases.

Creating a secondary database requires three things: a primary database, a binding for the keys, and a binding for the values. The value binding will be the same one used by the primary database. The key creator it must implement is the `SecondaryKeyCreator` interface, which has the single method `createSecondaryKey`.

Transactions

Transactions are enabled by an application when an environment and its databases are created or opened. Once enabled, they must be used for all database modifications. Methods such as `Database.get` and `Database.put` take a transaction object as an argument. If the transaction argument is null and transactions are enabled, the method may either use `autocommit` (committing the data when the operation is finished) as does “put” or it may not use a transaction at all, as with “get”, which does not modify the database.

The `TransactionRunner` class and `TransactionWorker` interface can make using transactions easier by handling retries and exceptions. A `TransactionRunner` creates a transaction and calls `TransactionWorker.doWork`. The runner can retry the work any number of times.

A Sample Application

A pair of electronic voting applications – one that simulates election day and another that runs reports against the collected data – will help illustrate the use of JE. While developing this application, I made heavy use of Sleepycat’s JE Javadocs and “Getting Started with

Berkeley DB Java Edition,” included in the documentation that comes with JE.

The two applications share much of their code. To create the data, the election day simulator creates voting booths, then generates votes and sends them to the vote server. The vote server stores information about the booths and their votes into multiple primary and secondary databases. The reporting application performs queries about the booths and the votes.

The three main classes for these applications are `Booth`, `Vote`, and `VoteServer`. A booth is identified by an IP address and knows what state, city, and district it is in. The booth assigns each vote a sequence number. A vote’s unique identifier is a combination of the booth’s IP address and the vote’s sequence number. A vote holds a few more fields like the race (President, State Assembly, Dog Catcher), political party (Democrat, Republican, Silly), and candidate.

After thinking about the queries I wanted to be able to run, I decided that in addition to the primary databases for `Booth` and `Vote`, I would need three secondary databases: for booths using state as the key, for votes using race as the key, and for votes using race plus political party.

Bindings

Before creating databases, bindings for `Booth` and `Vote` need to be created. For the primary databases, I chose to use custom binding classes. Listing 1 contains the two methods implemented by `BoothBinding`. The JE classes `TupleInput` and `TupleOutput` know how to read and write strings and intrinsics (ints, longs, etc.).

Bindings for the keys in the primary databases are unnecessary because the

keys are so simple: the booth database’s key is the IP address byte array that needs no translation, and the vote database’s key is a string from which we can retrieve a byte array.

Each secondary database needs a `SecondaryKeyCreator`. One of the vote secondary databases is indexed by race and party. Listing 2 shows the method that, given a vote and its primary key, fills in the secondary key `DatabaseEntry`.

Databases

When opening a database for writing, it is created if it does not exist. A flag is set that determines if the database is transactional. Listing 3 shows this process.

Storing Votes

The code in Listing 4 creates a transaction, calls the method that stores the booth and its votes, and commits the transaction. (Listings 4–9 can be downloaded from www.sys-con.com/java/sourcec.cfm.) This code commits without synchronizing the change to the log file, which is risky (the data is held in memory until the next sync) but faster.

`Booth` and vote data are written to the primary database. The secondary databases are updated automatically. The code in Listing 5 stores a vote into the vote primary database.

Reports

The `VoteServer` runs five different reports, each using a different technique.

Total Votes

The following code retrieves the total number of votes in the primary vote database.

```
// try/catch not shown
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Database db = dbEnvironment.getVoteDb();
Cursor cursor =
    db.openCursor(null, null);

DatabaseEntry foundKey =
    new DatabaseEntry();
DatabaseEntry foundData =
    new DatabaseEntry();
int numRecords = 0;
while (cursor.getNext(foundKey,
                        foundData,
                        LockMode.DEFAULT)
        == OperationStatus.SUCCESS)
    ++numRecords;
```

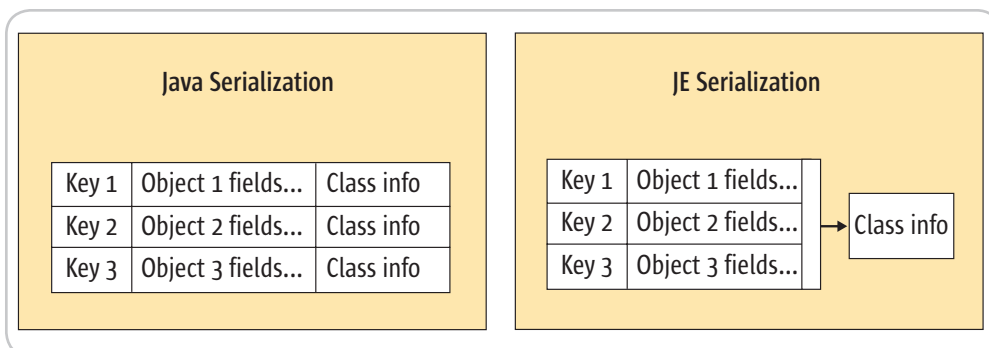


Figure 1 Serialization

```

cursor.close();
[DELETE:
int nRecs = dbEnvironment.getVoteDb()
    .getStats(null).getLnCount();
END DELETE]
// (print number of records here)

```

Single Vote

Database.get is used to find a single vote by its primary key (see Listing 6).

Booths by State

This report selects a state from the database, then prints all of the booths in that state. It runs the same query two different ways: with a cursor (see Listing 7) and with a JE Iterator (see Listing 8). The Iterator code is shorter. In both listings, error handling code and try/catch blocks are not shown.

Total Presidential Race Vote Count

To count how many votes were cast in the presidential race, a StoredMap is opened on the proper secondary database. The duplicate records (votes) for the presidential race are retrieved

and the number of votes is reported (see Listing 9).

Presidential Race Results

The last report prints the total votes per party for the presidential race and declares a winner. As in the previous report, a map is created on a secondary database and the map's size method returns the vote count.

Conclusion

The electronic voting application source code contains plenty of comments and deals with all the details ignored in this article such as error handling. The code and properties files that configure the databases and transactions have not been tuned for performance.

To get started with the Berkeley DB Java Edition, download it from Sleepycat Software. Read "Getting Started with Berkeley DB Java Edition," browse the example code that comes with JE, and refer to the JE Javadocs.

JE is available under a dual license.

The open source license is online at www.sleepycat.com/download/oslicense.html. Pricing starts at \$40,000 for a buyout license, which enables the customer to redistribute JE within a specific product/service in any volume into perpetuity. ☺

Resources

Sleepycat Software's site is www.sleepycat.com. There you can download the Berkeley DB Java Edition. The documentation and example code that comes with JE are great starting points. The site also contains a number of technical articles and white papers at www.sleepycat.com/company/technical.shtml.

The paper "Where's the Sleeping Cat?" is available at www.sleepycat.com/pdfs/index.php?paper=jdj_wheresthecat.

The latest version of the electronic voting application code can be found at www.io.com/~jimm/writing/evote.tar.gz.

Listing 1: BoothBinding

```

public void
objectToEntry(Object object,
    TupleOutput to)
    throws IOException
{
    Booth booth = (Booth)object;
    for (int i = 0; i < 4; ++i)
        to.writeByte(booth.address[i]);
    to.writeString(booth.state);
    to.writeString(booth.city);
    to.writeString(booth.district);
}

public Object
entryToObject(TupleInput ti)
    throws IOException
{
    byte[] address = new byte[4];
    for (int i = 0; i < 4; ++i)
        address[i] = ti.readByte();
    return new Booth(address,

ti.readString(),

ti.readString(),

ti.readString());
}

```

Listing 2: Creating a secondary key

```

public boolean createSecondaryKey(
    SecondaryDatabase db,
    DatabaseEntry keyEntry,
    DatabaseEntry dataEntry,
    DatabaseEntry resultEntry)
    throws DatabaseException
{
    // (Error handling code skipped)
    // The vote is in a dataEntry.
    Use
    // the dataBinding passed into
    this
    // SecondaryKeyCreator's con-
    structor
    // to convert it into a Vote
    object.

```

```

Vote vote = (Vote)dataBinding
    .entryToObject(dataEntry);

// Create a string key
String key = "" + vote.race +
':'
    + vote.candidateParty;

// Convert the key string to a
byte
// array and give it to result-
Entry,
// which is the DatabaseEntry
we're
// supposed to fill with the
secondary
// key.
resultEntry
    .setData(key.getBytes("UTF-
8"));

return true;
}

```

Listing 3: Creating and opening a database

```

private Database
openDatabase(boolean readOnly,
    String databaseName)
{
    DatabaseConfig config =
        new DatabaseConfig();
    config.setAllowCreate(!readOnly);
    config.setTransactional(!readO
nly);
    Database db = null;
    try {
        // If transactional, will use
        // auto-commit during the open
        db = dbEnvironment
            .openDatabase(null, databas-
eName,
                config);
    }
    catch (DatabaseException dbe) {
        // ...
    }
    return db;
}

```

Google Seeks Expert Computer Scientists

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

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





Joe Winchester
Desktop Java Editor



Turkish Java Needs Special Brewing

On a recent trip to Turkey to meet with a customer, I heard a comment that one of the reasons Java is being held back in that country is because of an almost ubiquitous locale bug.

In the Turkish alphabet there are two letters for “i,” dotless and dotted. The problem is that the dotless “i” in lowercase becomes the dotless in uppercase. At first glance this wouldn’t appear to be a problem; however, the problem lies in what programmers do with upper- and lowercases in their code.

The two lowercase letters are `\u0069` “i” and `\u0131` (dotless “ı”) and are totally unrelated. Their uppercase versions are `\u0130` (capital letter “I” with dot above it) and `\u0049` “I”. The issue is that this behavior does not occur in English where the single lowercase dotted “i” becomes an uppercase dotless “I.”

With the statement `String.toUpperCase()`, most Java programmers try to effectively neutralize case. Consider a `HashMap` with string keys and you have a key that you want to look up. If you want to ignore case, you’ll probably uppercase everything going into the map, its entries, and the string you’re doing the lookup with. This works fine for English, but not for Turkish, where dotless becomes dotless. I was shown an example of this bug in a popular HTML editor where a developer had done this with the set of HTML tags, so `<title>` would be indistinguishable from `<TITLE>` to their program and all variants in between, and probably looked like:

```
if (tagEnteredByUser.toUpperCase().
equals("TITLE")){
doTitleTagStuff();
}
```

In Turkish when “title” is entered, the resulting uppercase string has a dotted uppercase I (not the English dotless one) and the program wasn’t working as desired. This bug is just one example of where it had occurred. Another popular Java application failed with a similar bug tied back to the following code:

```
if (System.getProperty("os.name").toUpperCase().
equals("WINDOWS")){
doStuffSpecificForWindows();
}
```

The current locale is set as the user’s country, and the implementation of string methods use the default locale.

```
String toUppercase(){
return toUppercase(java.util.Locale.getDefault());
}
```

Given that this works for English (where `/u0060` uppercases to `/u0049` correctly), why doesn’t it hold true for Turkish? The developer did find special code that deliberately does the dotted to dotted, dotless to dotless, complete with a comment ironically stating:

```
// special code for turkey
```

The solution is to specify an explicit English locale when uppercasing for programmatic purposes, so the first line of buggy code would become:

```
if (tagEnteredByUser.toUpperCase(java.util.
Locale.ENGLISH).equals("TITLE")){
doTitleTagStuff();
}
```

Even if this were diligently done by everyone developing your code, you’ll still encounter a problem when using something written by someone else whose source you don’t have access to. For this the current workaround by Tamar Sezgin and others is to switch the locale of the program before the buggy code, make the call, and then switch back.

```
Locale.setDefault(Locale.ENGLISH);
// Use incorrectly written code
Locale.setDefault(new Locale("tr",""));
```

The problem with this is that it fails to follow the principle of least astonishment. It’s only there because Java supports locale-sensitive case conversion.

However, this isn’t offered by alternatives such as VB, C++, or Delphi, where case conversion follows English rules and if you want to do dotless “correctly” you have to implement it yourself. The only case where you would actually want to do it “correctly” would be for a user-visible string accepting a Turkish name (such as a surname), and the developers who want to do this would be those who were more likely to be aware of locale issues. The exception would then be:

```
Locale turkishLocale = new
Locale("tr","");
String tag = anotherUserVisibleString.toUpperCase(turkishLocale);
String s2 = anotherUserVisibleString.toUpperCase(turkishLocale);
if (s1.equals(s2)){
doSomethingFunWithTwoEqualsStrings();
}
```

However, even better would be:

```
if (sq.equalsIgnoreCase(s2)){
doSomethingFunWithTwoEqualsStrings();
}
```

so the only real case of wanting to uppercase a user-visible string to compare against another user-visible string is left to developers of database indexes and doesn’t need to be tackled at all by most Java programmers.

There is a PMR 53119 open to try to get Java changed so the default logic is to assume the string is not user visible. However, because this would be a breaking change to the current behavior, it can’t be done. In the meantime, I would urge all developers who ever find themselves converting a string into upper- or lowercase to think about whether these are user-visible strings. If not, make sure you explicitly use the English locale, otherwise you’re going to serve up Java that tastes great everywhere except Turkey.

• • •

I would like to thank Tamar Sezgin of IBM Turkey for explaining this problem to me and helping with this editorial. ☺

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

joewinchester@sys-con.com



What//: are your Web applications saying to your customers?



Have you survived a Web App from Hell? Tell us how you lived through the delays, confusion, and pain, and H&W will put your name in the hat for an HP iPAQ H5500 Pocket PC.

Share your story today at:
www.hwcs.com/jdj03.html

You might as well be closed for business if your Web applications aren't running at top speed 24 hours a day. With poor Web performance costing businesses \$25 billion a year, you need to find problems, and you need to find them fast.

You need DiagnoSys.

DiagnoSys is the first intelligent performance management solution that analyzes your J2EE and .NET applications from end to end. DiagnoSys starts analyzing quickly, gathers data directly from all application resources, and proactively finds and helps fix the real cause of problems – before they cost your business big time.

So when it's absolutely essential that your Web applications say "Open 24 hours," trust DiagnoSys.

© 2003-2004 H&W Computer Systems, Inc.
DiagnoSys is a trademark of H&W Computer Systems, Inc.



www.hwcs.com | 1.800.338.6692

When Mars Is Too Big to Download

by Michael Jacobs

In my previous article ("Bringing Mars Down to Earth with Java3D," JDJ, Vol. 9, issue 6), readers were expected to download hundreds of megabytes of Mars data to enjoy the Java3D example. This requirement challenged even the cable modem bunch ambitious enough to get the source code in the first place. This time it's definitely different. This time, the code generates the landscapes so all you have to download is the source. We'll cover the foundational Java3D data structures suitable for terrains, and how to completely generate landscapes with fractals. (The source code for this article can be downloaded from www.sys-con.com/java/sourcec.cfm.)



Mike Jacobs is a technical architect working at the Mayo Foundation for Medical Education and Research. Mike has developed CPU hardware, microcode, application components, and applications in the financial and health care industries. He has extensive design and implementation experience in object-oriented languages including Smalltalk, C++, and Java.

jacobs.michael@mayo.edu

Laying the Foundation

When you first started programming in Java, you may have started with the infamous "Hello World" example to introduce yourself to the basics of Java programming. In similar fashion, this article covers the basics of creating a Java3D terrain world and explores the possible approaches to creating that world.

Modeling Height Fields

No terrain article would be complete without at least a passing reference to height fields. Just in case you're new to terrain rendering, height fields or height maps represent a regular grid of longitude, latitude, and altitude values for a landscape. Interpreting these values and rendering them quickly with the right colors and lighting is at the core of rendering terrains (see Figure 1).

Each cell corner in the grid represents an altitude at a longitude and latitude location on a map. These values can be used to create three-dimensional points in the Java3D

world. The longitude is mapped to the x -axis, the latitude to the z -axis, and the altitude to the y -axis. We can use a QuadArray as our first simple approach to modeling a height field in Java3D.

The QuadArray Class

The QuadArray class is one of several GeometryArray subclasses provided by Java3D to create geometries. The QuadArray class is for rendering a group of quadrilaterals by specifying the corners of each quadrilateral. This is exactly how we described a height field earlier. The secret to using a QuadArray class is how you specify the corners of the quadrilaterals (see Figure 2).

The cell corners in the height field grid can be implemented as the corners of the quadrilaterals in the QuadArray. The corners must be specified in a counterclockwise order. The first corner is not important, and neither is the order of the quadrilaterals. The HelloWorld example code starts in the lower left-hand corner for both. The result of running the HelloWorld example is shown in Figure 3.

The examples in this section use hard-coded values for the height field and are implemented in the getHeightField() method. The geometry-building code in the HelloWorld example is shown in Listing 1.

This example uses the GeometryInfo utility class to make using the QuadArray easier. The inner loop assigns the coordinates of one square of the height field in a counterclockwise order while also mapping the points to the Java3D coordinate system.

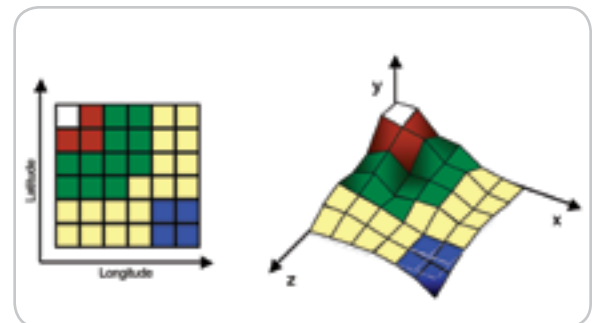


Figure 1 Interpreting a height map as a terrain

This approach is fine for learning but has a few drawbacks.

1. A grid corner can be included in the coordinates array up to four times as separate copies of the same point. This approach is fine for small landscapes but will burn through memory for larger terrains. Solving this with indexed geometry arrays will be covered later in this article.
2. While some terrain complexity reduction algorithms use quadrilaterals, most use triangles. Since we want to eventually get to build large-scale terrains and your 3D video card is optimized to render triangles, we should start thinking in terms of triangles.
3. The Java3D specification states that quadrilaterals must be planar or the results are undefined. The example makes no attempt to make the quadrilaterals planar and it might be difficult to make natural-looking terrains with them. While I haven't experienced any problems using nonplanar quadrilaterals, triangles are a safer route in the long term.

The TriangleArray Class

The TriangleArray class is another GeometryArray subclass provided by Java3D to create geometries. The TriangleArray class is for rendering a group of triangles by specifying the corners of each triangle (see Figure 4).

Once again, the starting corner and the order of the triangles are not important. The corners must be specified in a counterclockwise order. The HelloWorld2 example implements the same terrain as HelloWorld but uses a TriangleArray with the GeometryInfo class.

The geometry-building code in the HelloWorld2 example is shown in Listing 2. (Listings 2–8 can be downloaded from www.sys-con.com/java/sourcec.cfm.) The inner loop assigns the coordinates of two triangles for each square of the height field similar to the previous example. Note that the number of coordinates is higher with a TriangleArray compared to a QuadArray for the same regular grid of height data. While we want to use triangles, this approach uses even more memory than the QuadArray.

The TriangleStripArray Class

The TriangleStripArray class is an interesting GeometryArray subclass provided by Java3D to create geometries. The TriangleStripArray class is for rendering a series of triangles that share edges in a grouping called a strip. A strip of triangles is formed by specifying the corners of a triangle based on the last two corners of the previous triangle. A picture really helps to understand this one, so refer to Figure 5.

The first triangle is formed with corners (1,0), (0,0), and (1,1). The second triangle shares the hypotenuse with the first triangle and is formed with corners (0,0), (1,1), and (0,1). Notice how the last two corners of the first triangle are the same as the first two corners of the second triangle? Java3D takes advantage of this pattern and shares the corners so that only one additional corner is required to specify the next triangle. In the HelloWorld3 example, this continues across the row until the right edge of the height field. At this point the series of corners have specified a strip of triangles and the process is repeated for the next row.

The geometry-building code in the HelloWorld3 example is shown in Listing 3. Using a TriangleStripArray requires ad-

ditional information about the strips. Java3D needs to know the number of strips and the number of corners included in each strip. The number of strips is based on the size of the strip count array and the values in the array are the number of corners.

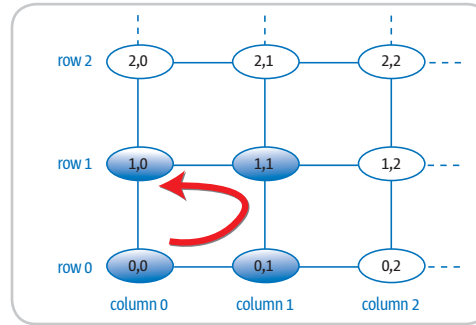


Figure 2 Specifying QuadArray coordinates

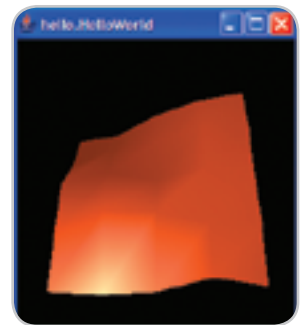


Figure 3 HelloWorld example

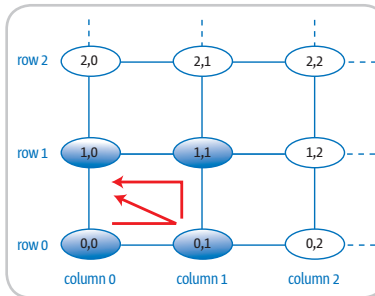


Figure 4 Specifying TriangleArray coordinates

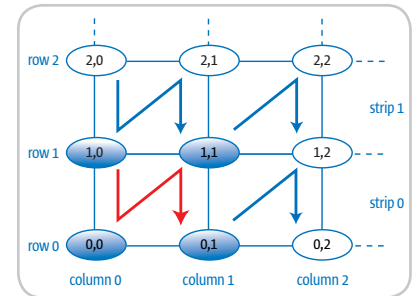


Figure 5 Specifying TriangleStripArray coordinates

WHO'S DEVELOPING THE COOLEST WIRELESS APPLICATIONS? YOU ARE!

ENTER TO WIN THE 2005 SIMAGINE DEVELOPERS' CONTEST!

Over \$70,000 will be awarded for innovative SIM card services including a special Cingular award for best submission from North America

Deadline is October 10, 2004!

In association with:

Full contest details at: www.simagine.axalto.com or call | 888 343 5773
© Axalto 2004

An advantage of using a `TriangleStripArray` is that it uses less coordinate memory than the other geometry array approaches. In addition, OpenGL is able to directly render a triangle strip array with a single call to the video card. DirectX must render each triangle individually, requiring multiple trips to the video card.

Using a `TriangleStripArray` does not require that the strips be organized as in the example. This was done simply for convenience and simplicity. As long as adjacent triangles share corners, you can theoretically create one long strip for your entire landscape.

The `TriangleFanArray` Class

Java3D provides one last option called the `TriangleFanArray`. There are several terrain simplification algorithms that substantially reduce the number of triangles used to render the landscape. The reduction is achieved by varying the size of the triangles to make the view “good enough,” as measured by different metrics. While these algorithms are beyond the scope of this article, it should be noted that the `TriangleFanArray` is especially well suited for rendering multiresolution triangulations.

Reducing Memory Burn

All of the geometry arrays explored thus far have used instances of `Point3f` to specify coordinates and they often represent the same 3D point several times. This approach could be optimized for large terrains if we could eliminate the use of `Point3f` and share the 3D points somehow.

Java3D includes additional versions of the geometry arrays called indexed geometry arrays. Once you understand how to use regular geometry arrays, the indexed counterparts are easy to understand. The indexed geometry array lets you specify the corners of the height field once and refer to them through an index. The index is used to specify which corner to use for the triangle or triangle strips (see Figure 6).

For this variety of index geometry arrays, the memory savings is realized at the shared corners where the strips touch. `HelloWorld4` is a version of `HelloWorld3` converted to use an `IndexedTriangleStripArray`. This example also eliminates the use of `Point3f` objects to further save memory.

The geometry-building code in the `HelloWorld4` example is provided in Listing 4. In this example, the coordinates do not use `Point3f` objects but a float value for each of the x , y , and z values. A loop populates the coordinate array with the values for each corner in the height map. Next, the indices array is populated with the index of the corner in the coordinates array. Using an index eliminates duplicate point data. The final difference is the setting of the coordinate indices on the `GeometryInfo` object.

The Java3D API provides several options for modeling height fields. Using indexed geometry arrays saves memory and triangle strip arrays lets your video card render your landscape quickly.

Fractal World, Fractal World, Excellent!

You have probably heard of fractals and may have found them a bit mysterious or unapproachable. I’ll demystify fractal concepts and show how you can shape them into realistic terrains (see Figure 7).

Overview

Since Benoit Mandelbrot discovered fractals over 20 years ago, many variations have been described in research papers and textbooks. The computer graphics community has come to use the term to mean anything that has a high degree of self-similarity. A self-similar object can be translated, rotated, and scaled to a subportion of itself. This concept is best described by an example called the von Koch snowflake (see Figure 8).

The von Koch snowflake is created by repeatedly replacing each line segment with a scaled-down version of the previous step. This is done by replacing each line segment in the current step with an exact copy of the previous step, scaled down by a factor of three. This process is repeated a number of times to create a self-similar snowflake. Replacing a portion of the shape with a scaled version of the previous step is the key concept behind fractals. How can fractals be applied to terrains?

Compare the ragged edges of a rock and the shapes of cliffs and mountaintops and you may observe self-similarity in nature. Because many natural forms such as plants and rocks seem to be self-similar, fractals have been used as a way to model these forms. Several fractal approaches have been used that all roughly fall into a category called fractional Brownian motion (fBm):

- **Poisson faulting:** The original terrain rendering approach subsequently improved by other approaches
- **Fourier filtering:** A complex interpretation of a Fourier transform of Gaussian white noise
- **Successive random additions:** A flexible and easy-to-implement subdivision scheme
- **Midpoint displacement:** An easy-to-understand and implement approach described in detail below
- **Noise synthesis:** The state of the industry in terrain generation

We’ll discuss midpoint displacement in this article. See the references for more information about the other approaches.

Midpoint Displacement

Fournier, Fussell, and Carpenter developed a way to generate fractal mountains based on a recursive subdivision. To better understand this approach, we’ll first go over the one-dimensional example shown in Figure 9.

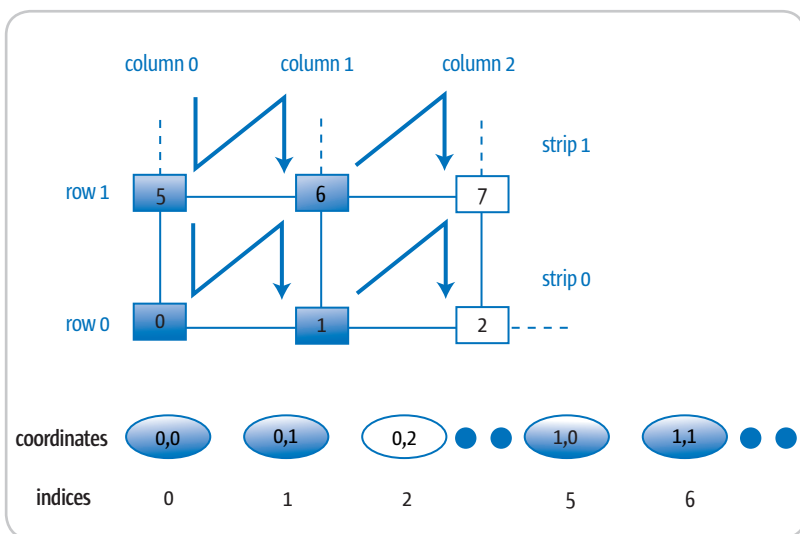


Figure 6 Specifying `IndexedTriangleStripArray` indices

We start with a line segment of unit length along the x -axis. In the second step, we divide the line segment into two equal halves and move the mid-point in the y direction. There are two line segments and in the next step they are divided and their mid-points moved in the y direction (up or down). This process is repeated a number of times to achieve the desired effect. How much should the midpoints be moved? The algorithm determines this by averaging the y values of the line segment end points and adding a random perturbation:

$$x_{new} = \frac{1}{2}(x_i + x_{i+1}), \quad y_{new} = \frac{1}{2}(y_i + y_{i+1}) + P(x_{i+1} - x_i)R$$

where $P()$ is a perturbation based on the length of the line segment and R is a random number between zero and one. In our case, the perturbation is called the roughness of the terrain. As long as the iterations gradually reduce the roughness, we can achieve self-similarity in the result. Applying this midpoint displacement algorithm to height fields allows us to create fractal mountain ranges. An

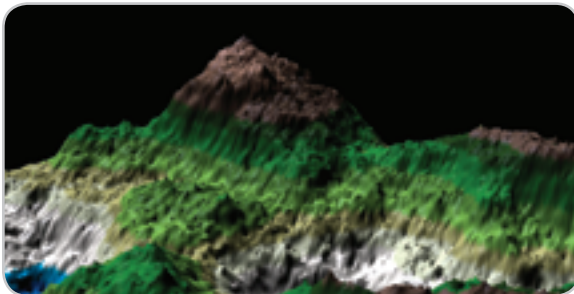


Figure 7 A fractal terrain rendered with Java3D



Figure 8 The construction of the von Koch snowflake

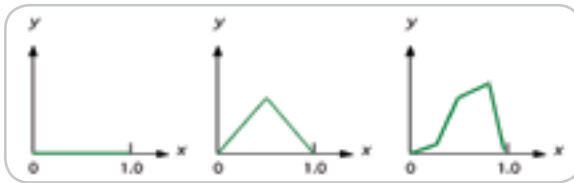


Figure 9 A one-dimensional subdivision example

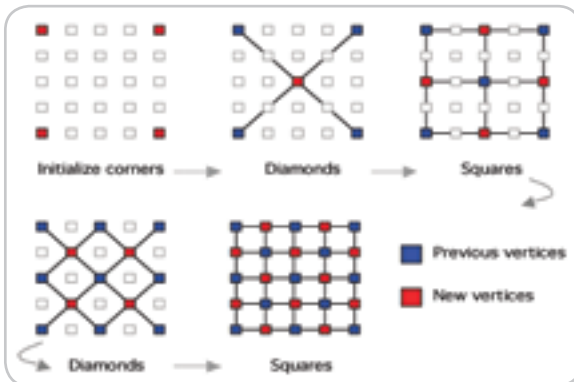


Figure 10 Steps of the diamond-square algorithm

extension of this algorithm is called the diamond-square algorithm.

The Diamond-Square Algorithm

The diamond-square algorithm gets its name from the imaginary shapes that result from the iterative midpoint displacement. This will become apparent as we walk through the algorithm. This terrain algorithm is similar to many in that it depends on the ability to split a line and have the result land squarely on a corner in the height field. To accomplish this feat, we must restrict the height field to be a square having $2^n + 1$ corners per side. The value of n determines how many iterations of midpoint displacement are possible, so we call this value the level of detail. A level of detail of three requires nine corners per side of the height map. For our example a level of detail of two will suffice for simplicity.

The steps in the diamond-square algorithm are:

1. Initialize the roughness.
2. Initialize the height field corners to form an imaginary square.
3. For each level of detail, do the following:
 - **Diamonds step:** Assign a height to the center of each square by averaging the height of each corner and displacing the result based on the roughness and a random number. This results in imaginary diamond shapes.
 - **Squares step:** Assign a height to the center of each diamond by averaging the height of each corner and displacing the result based on the roughness and a random number. This results in imaginary square shapes.
 - Scale down the roughness.

Figure 10 depicts the results during this process for a level of detail of two. This process can be used to create a height field that in turn can be used to generate a terrain. The FractalWorld example implements the diamond-square algorithm.

A Java3D Example

The FractalWorld example generates random fractal terrains with the diamond-square algorithm. This example is largely based on previous examples using the `TriangleStripArray` so we won't cover the geometry aspect of this example. The main generation loop is contained in the `getHeightField()` method shown in part in Listing 5.

This code implements the steps described above considering the boundary conditions of the height field. The diamonds step is straightforward:

```
private void diamond(
    float[][] terrain,
    int x,
    int y,
    int side,
    float roughness) {
    if (side > 1) {
        int half = side / 2;
```

Key	Function
Down Arrow	Move backward
Left Arrow	Turn left
Alt-Left Arrow	Strafe left
Right Arrow	Turn right
Alt-Right Arrow	Strafe right
PgUp	Look down
Alt-PgUp	Move up
PgDn	Look up
Alt-PgDn	Move down
=	Return to the starting point

Table 1 Keyboard bindings

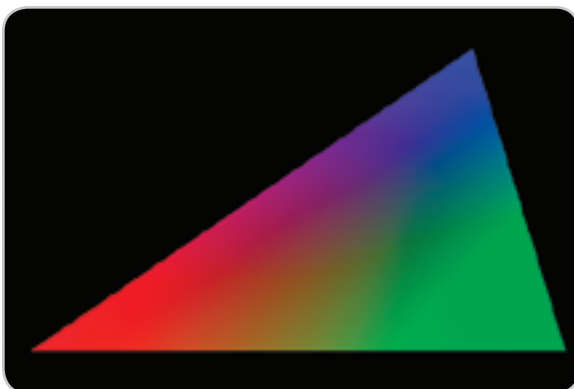


Figure 11 Vertex coloring on a triangle

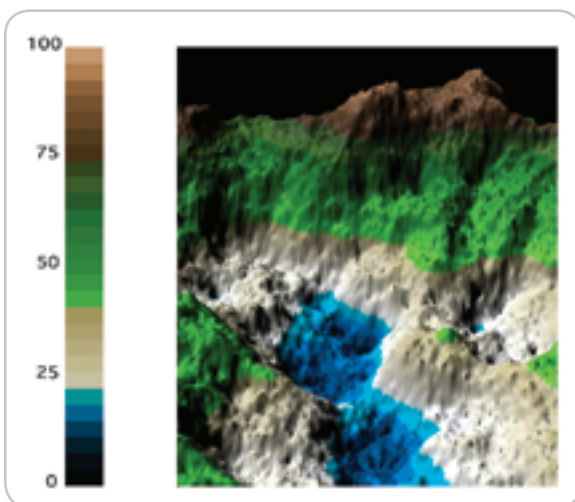


Figure 12 The vertex coloring used in the FractalWorld example

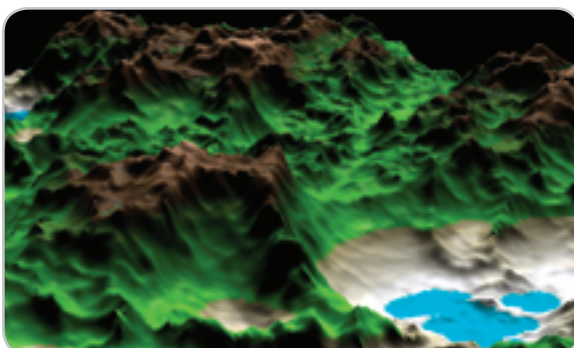


Figure 13 A multifractal landscape built with Perlin noise and Java3D

```

float sum = 0f;
sum += terrain[x][y];
sum += terrain[x + side][y];
sum += terrain[x + side][y + side];
sum += terrain[x][y + side];
float average = sum / 4.0f;
terrain[x + half][y + half] =
    average + random() * roughness;
}
}

```

The boundary conditions become more of a concern for the squares step since a diamond may extend beyond the physical dimensions of the height field (see Listing 6).

When a diamond corner is outside of the height field, this implementation simply does not consider it in the average calculation. Another option that I have seen in other implementations is to wrap the diamond to the other side of the height field and use a substitute corner in the average calculation.

Running the Example

When running the FractalWorld example, use the keyboard bindings shown in Table 1 to fly through the world.

The main() method sets up the roughness and level of detail. I've found that a roughness between 0.35 and 0.55 subjectively looks the best. The level of detail has a drastic effect on the results. A value of 8 seems to be a good balance between realistic results and speed. When running the example, you'll see colors on the landscape roughly corresponding to elevation. This is done with a Java3D feature called vertex coloring.

Vertex Coloring

In the HelloWorld examples, we used the ambient color in the material object to specify the color of the shape. This provides a convenient way to uniformly color a Java3D shape. Java3D also allows each corner of the triangles in the scene to have its own color. It combines the color of the material with the smoothly interpolated corner colors to produce the overall triangle color.

Figure 11 shows the results of running the SimpleVertexColoring example. Each corner is assigned a different color and Java3D takes care of the rest. The portion of the example that assigns the colors is shown in Listing 7.

This listing uses black as the ambient color of the triangle, making the vertex colors dominate. Note that the coordinates and colors use objects instead of indexes. As we discussed in Reducing Memory Burn, a large terrain should use indexed coordinates. Java3D also allows us to index colors in a similar manner and is demonstrated in the FractalWorld example.

Vertex Coloring in FractalWorld

This example uses elevation-based colors to approximate a more natural-looking landscape. The 27 colors were arrived at by trial and error with the help of a color utility. There is nothing significant about the number of colors I chose to use; I was looking for gradual changes from one elevation to the next.

A scene like Figure 12 has over 131,000 triangles so it's a good idea to conserve memory by indexing coordinates and colors. Colors are defined once and the color indices are built for each coordinate.

```
float[] colors = getElevationColors();
gi.setColors3(colors);
int[] colorIndices = getElevationColorIndices(hf);
gi.setColorIndices(colorIndices);
```

The `getElevationColors()` method defines the colors and they're set on the `GeometryInfo` object. Because the example uses a `TriangleStripArray`, the color indices are built similar to the coordinate indices (see Listing 8).

The values in the height field range from 0 to 100 (lowest to highest) and the colors are defined from highest to lowest. The `NUMBER_OF_COLORS` static variable is used to calculate the index of the color based on the elevation.

Conclusion

After running the `FractalWorld` example a few times you may notice that terrain roughness does not vary much. It would be nice if the beaches and hills were smoother than the mountains. If you are interested in solving the terrain roughness issues here are a few tips. The algorithm presented here is technically a monofractal, meaning that the fractal has a single uniform fractal dimension. An approach to varying the roughness of the terrain is called multifractals. You should have enough knowledge now to research multifractals and implement them in Java3D (see Figure 13). Have fun!

Acknowledgments

Thanks to my friend Jeff Ryan and my son Ryan for reviewing my *JDJ* articles. ☺

References

- Fournier, A.; Fussell, D.; and Carpenter, L. "Computer Rendering of Stochastic Models." *CACM*, 25(6), June 1982, 371-384.
- *The Virtual Terrain Project*: www.vterrain.org
- Ebert, D.S.; Musgrave, E.K.; Peachey, D.; Perlin, K.; and Worley, S. (2003). *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann Publishers.
- Ken Musgrave's doctoral dissertation: www.kenmusgrave.com/dissertation.html
- Sun's Java3D Home: <http://java.sun.com/products/java-media/3D/index.jsp>
- Java3D Resources: www.j3d.org/
- A useful color utility (*La Boîte à couleurs*): <http://pourpre.com/colorbox/indexen.php>

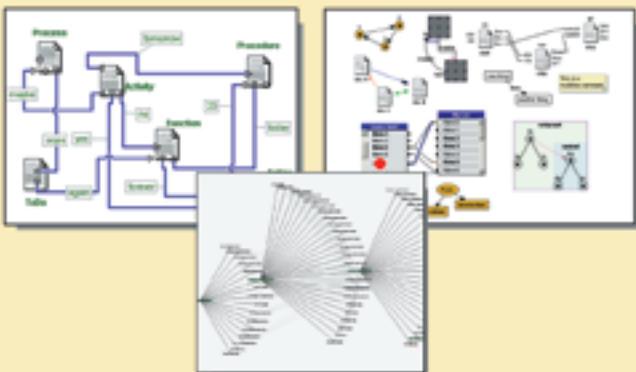
Listing 1

```
// The height field contains a float value for
// each [row, col] vertex.
float[ ][ ] hf = getHeightField();
BranchGroup objRoot = new BranchGroup();
GeometryInfo gi = new GeometryInfo(GeometryInfo.QUAD_ARRAY);
// The number of vertices (or coordinates) is based on the
// number of squares in the grid (4 by 4) times the number
// of corners (4) for each.
Point3f[ ] coordinates = new Point3f[64];
// For each corner of each square in the grid, convert the
// height field altitude value into instances of Point3f.
// The row is mapped to the minus z-axis, the column to
// the x-axis and the height field altitude value
// to the y-axis. Each iteration adds one square.
int ci = 0; // coordinate index
for (int row = 0; row < 4; row++) {
    for (int col = 0; col < 4; col++) {
        // use compass bearings to id the corners
        float sw = hf[row][col];
        float se = hf[row][col + 1];
        float ne = hf[row + 1][col + 1];
        float nw = hf[row + 1][col];

        coordinates[ci] = new Point3f(col, sw, -row);
        coordinates[ci + 1] = new Point3f(col + 1, se, -row);
        coordinates[ci + 2] =
            new Point3f(col + 1, ne, - (row + 1));
        coordinates[ci + 3] =
            new Point3f(col, nw, - (row + 1));
        ci = ci + 4;
    }
}
gi.setCoordinates(coordinates);
```

Build Incredible Interactive Diagrams


with **JGo™**



Create custom interactive diagrams, network editors, workflows, flowcharts, and design tools. For web servers or local applications. Designed to be easy to use and very extensible.

- Fully functional evaluation kit
- No runtime fees
- Full source code
- Excellent support

Learn more at:
www.nwoods.com/go
 800-434-9820



2004 RCAs

Readers' Choice Awards

Nominations Open

"The Oscars of the Software Industry"




Nominate the Best of the Best:
 Tell us what tools, solutions, services,
 and books deserve recognition.

Cast your vote at
www.SYS-CON.com

JDeveloper 10g

by Oracle Corporation

Reviewed by
Alain Trottier

When vendors start charging more than \$10,000 for a single tool, you know that the product category is about to heat up. Since Java IDEs have multiplied, I recently took Oracle JDeveloper 10g for a test drive – perhaps you didn't realize this vendor had a serious Java IDE.

Product Description

Oracle JDeveloper 10g is an integrated development environment (IDE) for building Java applications. It supports the latest industry standards for Java, XML, and SQL including J2EE 1.3, XML, WSDL, SOAP, UDDI, J2SE, and J2ME. Oracle JDeveloper's integrated features enable the developer to manage a project throughout the development life cycle of modeling, coding, debugging, testing, profiling, tuning, and deploying applications. These features are actually part of the product. JDeveloper offers enough features without cluttering the product with stuff I won't use. Contrast that with Microsoft: its Office suite is excellent, but I rarely use more than 10% of the features.

Installing and Using JDeveloper

I thought I must have skipped something – it took three minutes to install and run JDeveloper from a CD (JDK 1.4.1 is required). Get the JDeveloper zip file (from a CD or download it from <http://otn.oracle.com/software/products/jdev/>), unzip it to a new directory, then run it (Windows: [jdeveloper_root]\jdev\bin\jdev.exe and Other platforms: [jdeveloper_root]/jdev/bin/jdev). I wish all products installed so cleanly.

The tool is written in Java, and this version of Oracle JDeveloper has added many features. For example, it provides a visual layout editor for both HTML and Swing-based user interfaces. This won't replace Dreamweaver, but it's nice to have. The Data Control Palette window provides a view into the business services layer. The developer can bind user-interface components to a business service with a simple drag-and-drop from this

palette. Oracle has a lot of experience in this area due to its database heritage. Oracle JDeveloper embraces popular open source frameworks and tools, providing built-in features for Struts, Ant, JUnit, and CVS. For example, wizards provide an easy way to define test cases, test fixtures, and test suites for projects. Need a personal Java trainer? The CodeCoach feature in JDeveloper scans the application code and provides hints and tips on changes that can be made to optimize performance. You also get Code Metrics to evaluate the structure of the Java code by analyzing its complexity.

Another nice feature in Oracle JDeveloper is the page flow modeler. This modeler can be used for Struts, which has become a key part of enterprise Java shops. Of course, JDeveloper is the best Java IDE on the market for building Oracle DB applications. This is important, but since everyone talks about this, let's focus on the new features, especially the Software Development Life Cycle (SDLC) support and the Application Development Framework (ADF). Oracle touts ADF as the crown jewel of this release.

JDeveloper supports several steps of the SDLC. The ISO 12207, and the more readable IEEE 1074, standard defines the primary phases of the SDLC in academic detail (these need a reality check). Observing these standards, the leading IDE vendors have continually added support for an increasing number of SDLC phases. At first, IDEs added compiling, debugging, and source control (i.e., CVS menu items). Lately, project management, requirements, and QA have started to appear in IDEs. Borland and IBM invested heavily in SDLC functionality through acquisitions and integration. While late, Oracle's ADF now includes modeling, coding, debugging, testing, profiling, tuning, and deploying applications. JDeveloper is unique among enterprise class IDEs because the SDLC feature set is actually part of the product codebase, not plugins or acquisitions/integrations like its competitors. It's a seamless IDE and doesn't hog your hard drive.

Oracle Corporation

500 Oracle Parkway
Redwood Shores, CA 94065
Web: www.oracle.com
Phone: 650 506-7000

Specifications

Platforms: Any platform with JDK 1.4 support
Pricing: \$995

Test Environment

Dell Inspiron 8000, 1 GHz Intel Pentium III processor, 30GB Disk, 256MB RAM, Graphics RAM 32MB, Windows 2000 w/Service Pack 3

Product Snapshot

Target Audience: Java enterprise programmers
Level: Medium to advanced

Pros:

- Powerful J2EE development
- Simple installation
- Supports correct set of standards (Java, XML, SQL, UML)
- Life-cycle management
- Good online support and docs
- Company is fully committed to future of product

Cons:

- Should reduce steps to build/deploy EJBs
- Want more model-driven architecture



Figure 1 | JDeveloper application template editor

Development life-cycle support is more of a JDeveloper feature than an ADF feature, but ADF helps. Oracle ADF's main focus is on simplifying the J2EE development process through a visual and declarative approach. Oracle ADF is based on the Model-View-Controller (MVC) design pattern. ADF lets developers build a simple, yet full, MVC-based J2EE application without any coding – just drag-and-drop, set some properties, and you're done. Oracle doesn't claim ADF eliminates coding when building J2EE applications; developers

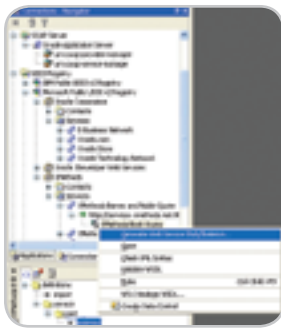


Figure 2 JDeveloper Web services wizard

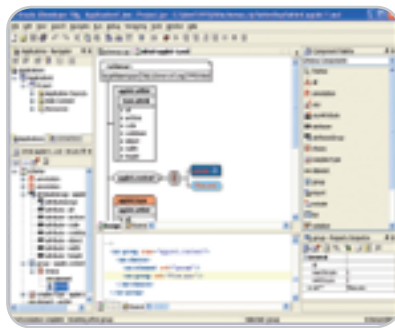


Figure 3 JDeveloper XML editor

would probably need to add some Java code. It does reduce the amount of coding for application infrastructures, letting developers focus on the business logic that is unique to their applications. The point of these wizards and the modeling aspects of the top IDEs is to automate routine coding chores. I really liked the application template feature shown in Figure 1.

The templates are cool because they accelerate development with ready-to-use J2EE design pattern implementations and metadata-driven components. I was relieved to see how simple it was to edit and play with templates. It really does anchor the high-level aspects of a design. JDeveloper templates approach the idea of model-driven architecture. Now the Object Management Group (OMG) has defined the MDA standard, which is targeted at providing a way to model, with UML, the complete application life cycle – design, deployment, integration, and management. Compuware OptimaJ does MDA best. I would like to see JDeveloper implement more of OMG's MDA in the near future. Presently, I really like how easy it is to skeleton an application in JDeveloper with these easy-to-pick/edit templates. Something else that caught my eye is that Oracle lets you choose your deployment platform for ADF. Many tools lock you into one application server. For example, BEA WebLogic Workshop only deploys to the BEA server and IBM's WebSphere IDE to WebSphere only. Even though Oracle has its own application server, you can deploy to all the major players including BEA, IBM, and JBoss. JDeveloper provides a visual XML Schema editor that lets XML developers browse XML Schemas easily. XML Schemas can be constructed visually using drag-and-drop from the component palette. Furthermore, by using XML-based industry standards, such as WSDL, SOAP, and UDDI, code components can be reused regardless of their location or the language used in their development. JDeveloper generates the necessary WSDL file to expose any Java class or PL/SQL package as a Web service, it supports the Web Services Interoperability standards, and can verify that Web services conform to the WS-I standards. JDeveloper make Web services easy to build, as shown in Figure 2.

The final feature I'll discuss is the XML editor. JDeveloper enables XML-based application development with features such as the XML Schema modeler, XML code insight, and the XML tag property inspector. An enterprise shop will still require a separate tool like XML SPY to do the heavy lifting, but the XML editor in JDeveloper feels good. It's strong enough to do the majority of the XML work, but it's not cluttered with extraneous features. It is easy to add/delete/modify tags, as shown in Figure 3.

Summary

Oracle has impressed me with their database products and their latest Java IDE is no exception. This product is designed for enterprise class projects. Of course, you can build GUIs and Web pages with it, but this tool's strength is the back end. This makes sense as JDeveloper's original motivation was to be a tool that was especially good at building Oracle DB applications in Java. If you're looking for a Java IDE with an attractive bang for the buck that makes it easy for you to manage the life cycle of enterprise J2EE and Web services applications, consider JDeveloper 10g. I used to think of JDeveloper as a tool that was married to the Oracle DB, but wasn't a real threat to the leading Java IDEs. Java continues to be a key component of Oracle's corporate strategy. Their commitment to JDeveloper is evident as 10g now joins the top IDE echelon. ☘

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	15
Apple	www.????.com	???-???-???	2-3
Axalto	www.simagine.axalto.com		53
Borland	www.go.borland.com/j6	831-431-1000	9
Business Objects	www.businessobjects.com/dev/p7	888-333-6007	19
Canoo Engineering AG	www.canoo.com/ulc	41 (61) 228 94 44	13
DataDirect	www.datadirect.com	800-876-3101	4
Dice	www.dice.com	877-386-3323	43
Freddie Mac	www.freddiemac.com	???-???-????	41
Google	www.google.com/cacm	650-623-4000	49
H&W Computer Systems	www.hwcs.com/jdj03.html	800-338-6692	51
Identify Software	www.identify.com		11
InetSoft	www.inetsoft.com/jdj	???-???-????	39
InferData	www.inferdata.com/jdimag	888-211-3421	33
InterSystems	www.intersystems.com/match1	617-621-0600	4
Jinfont	www.jinfont.com		35
Mindreef	www.mindreef.com	???-???-????	45
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	57
Oracle	www.????.com	???-???-????	27, 29, 31
Parasoft Corporation	www.parasoft.com/achievequality	888-305-0041 x3307	17
Quest Software, Inc.	http://www.quest.com/jdj	800-663-4723	Cover IV
Scientific Toolworks, Inc.	www.scitools.com		47
Sleepycat Software	www.sleepycat.com/bdbje	510-597-2128	25
Software FX	www.chartfx.com	800-392-4278	Cover III
Tangosol	www.tangosol.com		23
Web Services Edge 2005 East	www.sys-con.com/edge	201-802-3069	61

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.



Onno Kluyt

From Within the Java Community Process Program

Updating the first JSR

Welcome to the September edition of the JCP column! Each month you can read about the Java Community Process: newly submitted JSRs, new draft specs, Java APIs that were finalized, and other news from the JCP. This month we'll discuss the elections for the Executive Committees, three new JSRs for the J2ME technology, and news about the Community's first JSR.

Maintenance Review for JSR 1

In December of 1999, IBM got the JCP underway with its first JSR: the Real-Time Specification for Java. Since then the spec leadership changed hands to TimeSys Corporation. Now this JSR is back in the news with its recent completion of

Three New JSRs for the J2ME Technology

Originally submitted on June 1, JSR 246 (Device Management API) was approved by the ME EC after a so-called JSR Review Reconsideration Ballot. This proposed optional package for J2ME CLDC configurations, led by Siemens, provides a generic interface to the device management implementation in a device. It provides access to native device management protocols and to other functionality in the device such as triggering management sessions and change notification. JSR 246 is related to JSR 232 wherein the former's scope is CLDC and the latter's is CDC. Close collaboration between the spec leads is expected.

Vodafone and Nokia have jointly submitted JSR 248 (Mobile Service

JSR 185 for CLDC/MIDP-based devices, focusing specifically on high-volume handsets. Both JSRs are scheduled to complete by September 2005.

Early Draft Review

In JCP 2.6 all draft spec reviews are publicly accessible, including what used to be the Community Review and is now called Early Draft Review. One such review is currently underway for JSR 223, Scripting Pages in Java Web Applications. The JSR makes it possible for scripts to access and manipulate Java objects and for scripting pages to be used by Java server-side applications.

The EC Elections Are Coming Up!

Autumn is near and so are the yearly elections for the two Executive Committees. But before delving deeper I say "Thank you!" to Richard Monson-Haefel. Richard was elected last year to the SE/EE EC as an individual member. Richard has since found employment with an analyst firm and felt that continuing as an individual member on the EC created the potential for a conflict of interest; a decision we must respect. Richard's seat will remain empty until the elections. For the ME EC three nominated seats are coming up: Insomnia, RIM, and Sony; and two elected seats: Intel and Texas Instruments. For the SE/EE EC the three nominated seats that are expiring are Apache, Borland, and SCO; and the elected seats are Macromedia and Nokia Networks. The ratification vote on Sun-nominated seats begins on October 1 and the open election begins on November 1. Terms are three years with no limit on the number of terms.

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

“Autumn is near and so are the yearly elections for the two Executive Committees”

a maintenance draft review with the accompanying maintenance release coming shortly. The goal of the review is to clarify many portions of the spec and to make minor API changes to correct aspects of the RTSJ that were unstable.

The Java Virtual Machine Specification

A second JSR that is in the maintenance review stage is JSR 924. It highlights the changes that are needed as part of the imminent J2SE 5.0 platform release, calling out several changes to class file formats, instruction set, and loading and linking.

Architecture for CLDC) and also JSR 249 (Mobile Service Architecture for CDC). JSR 185 set a baseline for what a complete collection of Java technology for mobile devices should look like. As such it focused on CLDC/MIDP environments. JSR 249 plans to do the same thing but is now focused at the very high end of the market, the most capable devices currently available, and is using the CDC configuration with the Foundation Profile as a starting point. The expert group plans to deliver a specification for such devices and a road map document providing the future technical direction. Closely related to this effort is, of course, JSR 248, which picks up the course set by

Onno Kluyt is the director of the JCP Program Management Office, Sun Microsystems.
onno@jcp.org



web services **EDGE**
conference & expo

Web Services Edge 2005 East

International Web Services Conference & Expo

**Announcing Web Services Edge 2005
Extending Call for Papers to August 30th**
Submit your proposal today!

The Largest
i-Technology
Event of
the Year!



**Guaranteed
Minimum
Attendance
3,000
Delegates**



Tuesday, 2/15:
Conference & Expo

Wednesday, 2/16:
Conference & Expo

Thursday, 2/17:
Conference & Expo

**Hynes Convention Center, Boston, MA
February 15-17, 2005**

Join us in delivering the latest, freshest, and most proven Web services solutions... at the Fifth Annual Web Services Edge 2005 East - International Conference & Expo as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics including:

- Transitioning Successfully to SOA
- Federated Web Services
- ebXML
- Orchestration
- Discovery
- The Business Case for SOA
- Interop & Standards
- Web Services Management
- Messaging Buses and SOA
- SOBAs (Service-Oriented Business Apps)
- Enterprise Service Buses
- Delivering ROI with SOA
- Java Web Services
- XML Web Services
- Security
- Professional Open Source
- Systems Integration
- Sarbanes-Oxley
- Grid Computing
- Business Process Management
- Web Services Choreography

3-Day Conference & Education Program features:

- Daily keynotes from companies building successful and secure Web services
- Daily keynote panels from each technology track
- Over 60 sessions and seminars to choose from
- Daily training programs that will cover Web Service Security, J2EE, and ASP.NET
- FREE full-day tutorials on .NET, J2EE, MX, and WebSphere
- Opening night reception

**Interested in Exhibiting,
Sponsoring or Partnering?**

Becoming a Web Services Edge Exhibitor, Sponsor or Partner offers you a unique opportunity to present your organization's message to a targeted audience of Web services professionals. Make your plans now to reach the most qualified software developers, engineers, system architects, analysts, consultants, group leaders, and C-level management responsible for Web services, initiatives, deployment, development and management at the regions best-known IT business address - The Hynes Convention Center in Boston.

For exhibit and sponsorship information please contact Jim Hanchrow at 201.802.3066, or e-mail at jimh@sys-con.com.

Sponsored by:



All brand and product names mentioned above are trade names, service marks or trademarks of their respective companies.

Contact for Conference Information: Jim Hanchrow, 201-802-3066, jimh@sys-con.com



www.sys-con.com/edge

One Man's Open Source, Another Man's Asset

by Henry Roswell

There seems to be a lot of activity surrounding Java and open source. Simon Phipps, chief technology evangelist for Sun, threw down the gauntlet to IBM with his keynote at EclipseCon. He said that IBM (who Simon once worked for) already has at least four of their own open source implementations of Java and rather than everyone beating up on Sun to open source Java, perhaps IBM should step up to the plate with one of their implementations.

Following this, IBM's VP of emerging technology Rod Smith sent an open letter to Sun chief engineer Rob Gingell saying how they would answer Simon's challenge and work with Sun to create an independent open source imple-

community (Open Office, Network File System, etc....) so they already practice open source. And Java works fine with the JCP.

While Jonathan's arguments do make certain sense, Scott McNealy unfortunately didn't help the debate with his flat rebuttal to IBM: "Go open source DB2 and then you can tell me what to do with my assets" (www.linuxworld.com/story/44208.htm). To many who sat on the fence up to this point, his comments just added fuel to the fire that Sun likes Java being theirs and are playing dog in the manger.

More confusion came as Raghavan Srinivas, a Java technology evangelist from Sun, stated, "We haven't worked out how to open source Java – but at

weight this year, especially given that Sun and Redmond have recently kissed and made up. "Make no mistake: we will open source Solaris," Jonathan Schwartz insisted, hence upping considerably the amount of Sun license capital that they are now contributing to open source (<http://weblogs.java.net/pub/wlg/1543>).

Sun's problem may be that while they think Microsoft will ride into town and fragment Java, ironically it may actually be their own stance that achieves this. The recent SDO work by IBM and BEA was done totally out of the JCP community and runs counter to Sun's own JDO initiative. There are already open source initiatives for Java such as GNU (www.gnu.org/software/java/)

“Sun's problem may be that while they think Microsoft will ride into town and fragment Java, ironically it may actually be their own stance that achieves this”

mentation of Java. Rod stated: "Simon's comment appears to be an offer to jointly work toward this common goal. IBM is a strong supporter of the open source community, and we believe that a first-class open-source Java implementation would further enhance Java's position."

After Simon seemingly got his wish, Jonathan Schwartz disappointed everyone with his reiteration that open source would be the death of Java. His rationale is that Microsoft will roll into town and fragment Java as they had already tried, and even without them the open source model leads to market fragmentation as has occurred with Linux derivatives. Besides, Sun already gives a lot of stuff to the open source

some point it will happen...it might be today, tomorrow, or two years down the road" (www.zdnet.com.au/news/software/0,2000061733,39149502,00.htm).

The arguments flared up again at JavaOne when Rob Gingell referred to the IBM open letter as "corporate terrorism." Apart from the obvious lack of sensitivity shown to the subject he was using as his metaphor, it didn't help to initiate any kind of rational discussion. Later, in the debate led by Tim O'Reilly, it was pointed out that all the arguments given by Sun against open sourcing Java were just general arguments against any kind of open source. The Microsoft argument used at JavaOne didn't carry quite so much

[java.html](http://www.java.html)) and the Kaffe JVM (www.kaffe.org). Eclipse is open source and has overtaken NetBeans and fragmented the tools space; SWT is open source and has fragmented the Java desktop space.

What Sun should do is set up a separate company owned by the JCP members; this company should steer Java but operate outside of Sun's commercial interests (i.e., fair use of Java trademark, references to NetBeans in JRE press releases, fair licensing of Java technology, shared IP ownership). They won't, of course, but the bitter irony is that this is just what IBM did with the Eclipse foundation after Sun repeatedly refused to join, saying it was too blue for them. ☹

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

henry@sys-con.com

FREE JAVA CHARTS!

Download the Chart FX for Java Community Edition now.



www.chartfx.com



SPEND LESS TIME PROBLEM SOLVING... AND MORE TIME DEVELOPING APPLICATIONS.



PerformaSure – a system-wide performance diagnostic tool for multi-tiered J2EE applications running in test or production environments.



JProbe – a performance tuning toolkit for Java developers.

Join The Thousands of Companies Improving Java Application Performance with Quest Software.

Whether it's a memory leak or other performance issues, Quest Software's award-winning Java products — including JProbe® and PerformaSure™ — help you spend less time troubleshooting and more time on the things that matter. Quest's Java tools will identify and diagnose a problem all the way down to the line of code, so you no longer have to waste time pointing fingers or guessing where the problem lies. Maximize your team's productivity with Quest Software by downloading a free eval today from <http://www.quest.com/jdj>.



© 2004 Quest Software Inc., Irvine, CA 92618 Tel: 949.754.8000 Fax: 949.754.8999