SYS-CON

HOW DOES THE SUN–MICROSOFT SETTLEMENT AFFECT JAVA? PG 10

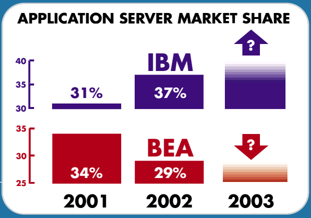
No. 1 *i*-Technology Magazine in the World

JDJ

MAY 2004 VOLUME:9 ISSUE:5

IN THIS ISSUE...

NO.1 CIRCULATION 140,000 IN THE WORLD!



PLUS...

– THE – FUTURE OF MIDDLEWARE & TOOLS

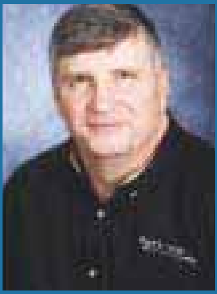
JDJ Exclusive: Meet the GMs of IBM's Software Group

Robert LeBlanc

John A. Swainson

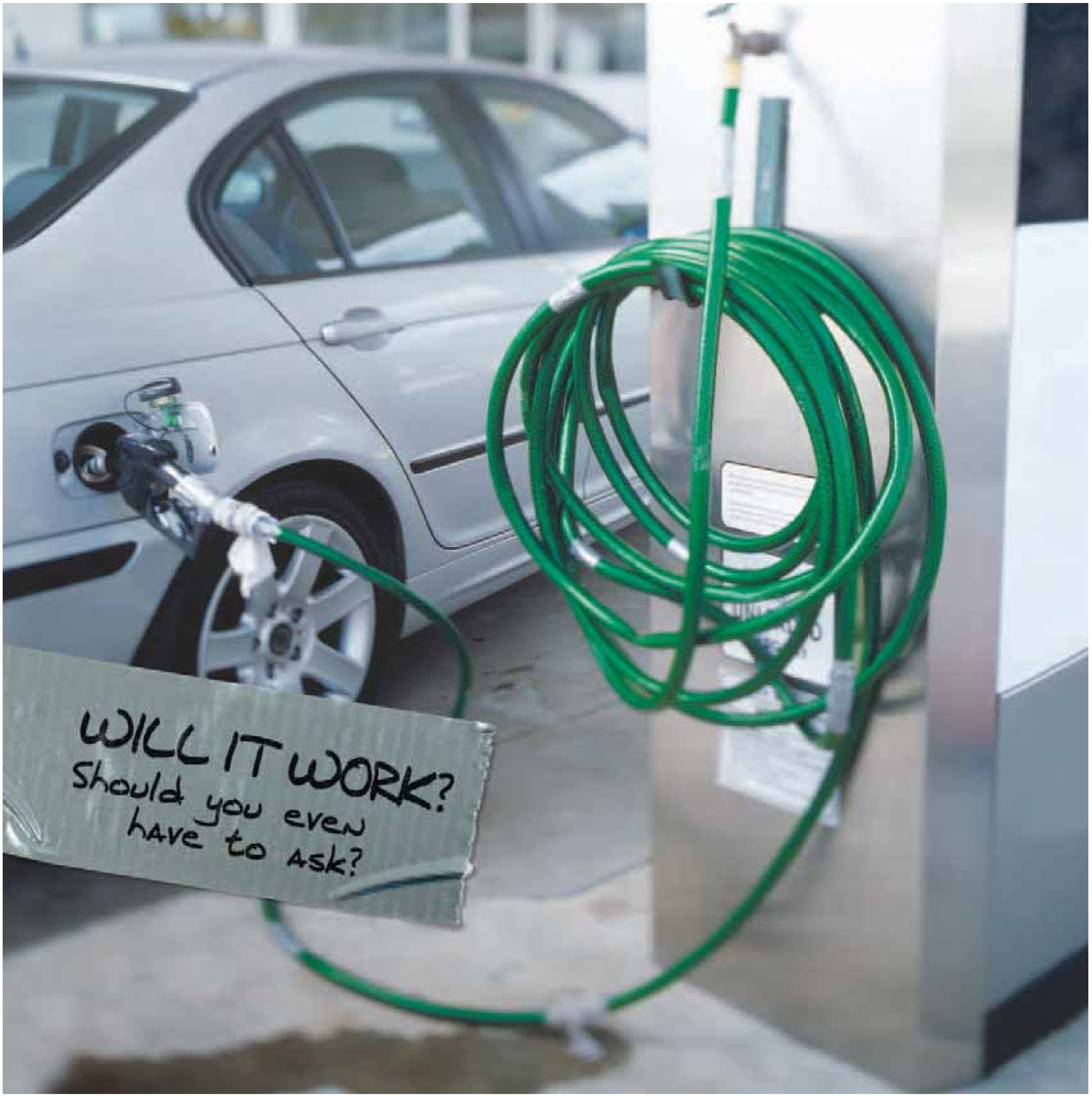
Janet Perna

Ambuj Goyal


Mike Devlin

RETAILERS PLEASE DISPLAY
UNTIL JULY 31, 2004

$9.99US $9.99CAN

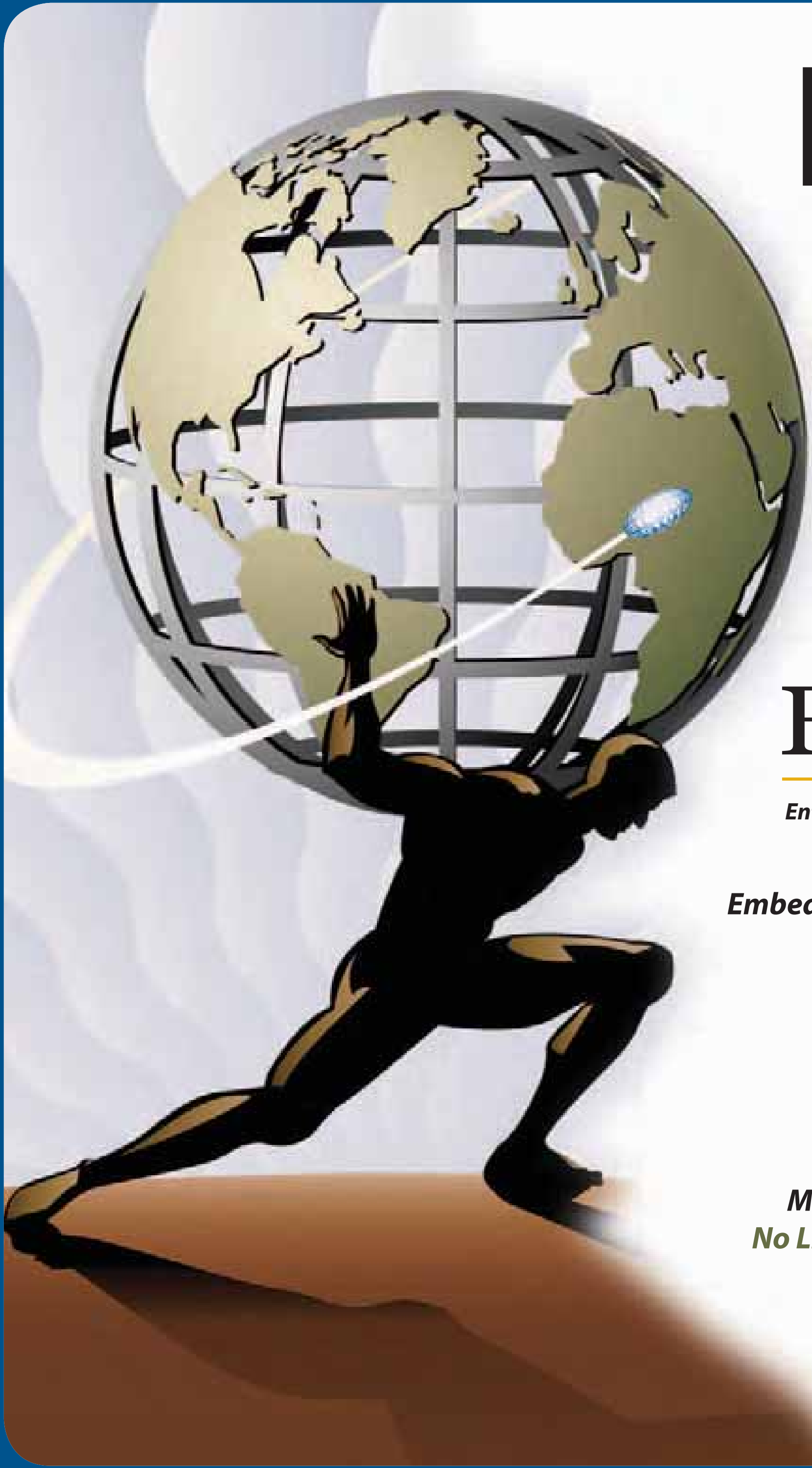


Don't take chances connecting your application to your data. Rely on DataDirect™ for premium JDBC drivers with support for advanced functionality like distributed transactions, connection pooling, and BLOB/CLOB. Our Type 4 drivers are fully database independent and are the SPECjAppServer/ECPerf performance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

DataDirect™ TECHNOLOGIES

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

www.datadirect.com 800-876-3101



BUILD JAVA REPORTS WITH NO LIMITS

FORMULA ONE®

Enterprise Reporting Toolsets for Java Projects & Portals

Embeds Into and Leverages J2EE Environments!
No Limits on IDEs or Application Servers!

UNLIMITED *productivity*
UNLIMITED *data access*
UNLIMITED *report output*

Most Flexible License! Extraordinary Value!
No Limits on Savings for Development Teams!

UNLIMITED *end users*
UNLIMITED *projects*
UNLIMITED *server CPUs*

PRODUCTIVE DEVELOPERS • SUCCESSFUL PROJECTS • EMPOWERED USERS

ReportingEngines™
JAVA REPORTING TOOLS FROM ACTUATE

FREE 30 DAY TRIALS

WEB: http://www.ReportingEngines.com
EMAIL: sales@ReportingEngines.com
TEL: 913-851-2200 • 888-884-8665
FAX: 913-851-1390

FORMULA ONE E.REPORT ENGINE (EMBED PDF, HTML, DHTML, AND XML REPORTS IN JSP AND SERVLETS)
Build reports against Java objects, JDBC, and XML. No report server to set up or maintain.

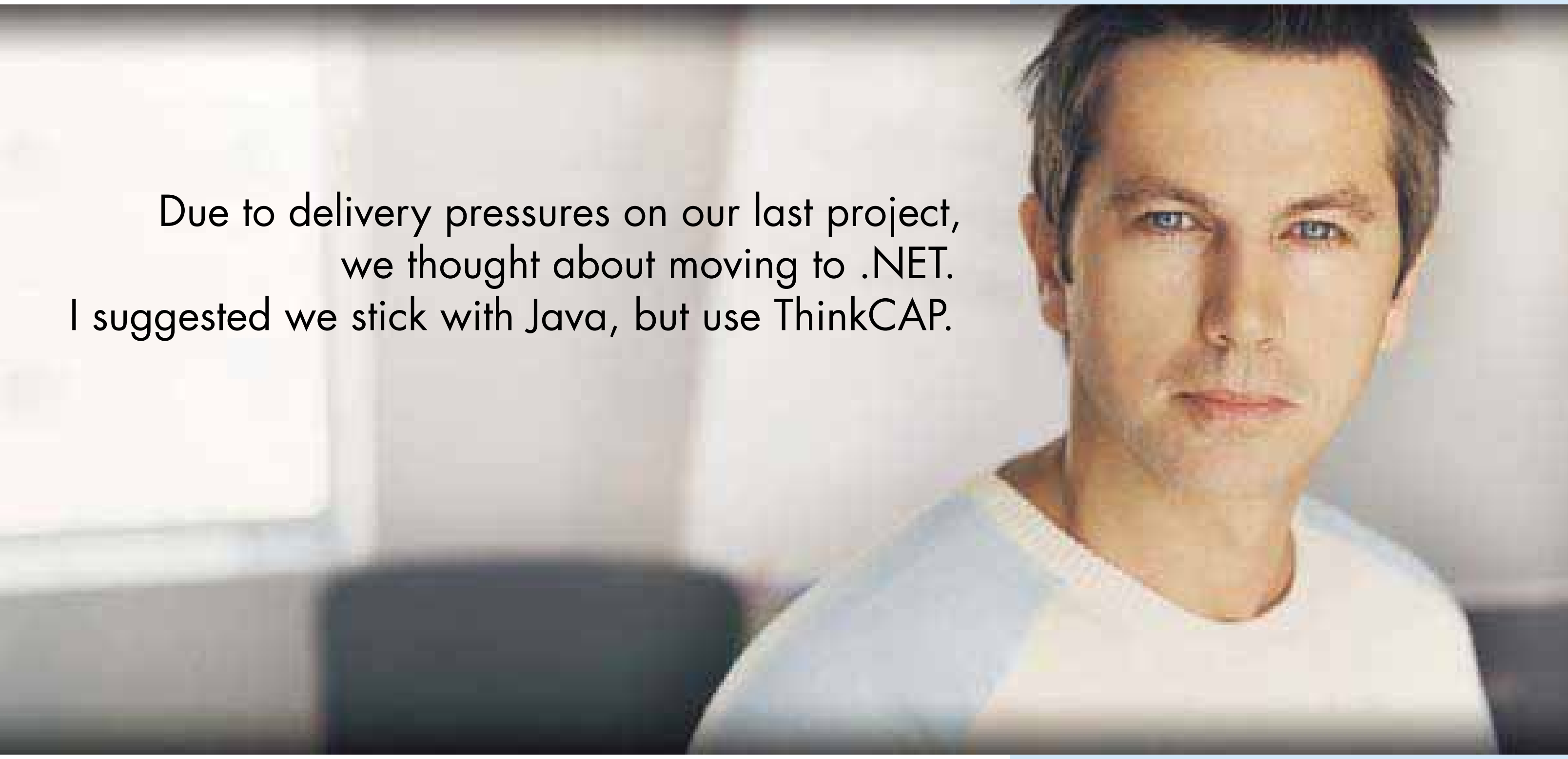
FORMULA ONE E.REPORT ENGINE FOR WEBLOGIC WORKSHOP (ACCESS LIQUID DATA AS A DATA SOURCE)
All the power of the regular e.Report Engine as a fully integrated BEA Workshop Extension.

BEA PORTAL REPORTING SOLUTIONS (POWERED BY THE FORMULA ONE E.REPORT ENGINE)
Visually build reports against BEA Portal Event and Behavior Tracking Data.

FORMULA ONE E.SPREADSHEET ENGINE (100% PURE JAVA TOOL FOR BUILDING FINANCIAL APPLICATIONS)
Excel reporting. Excel-like data grids. Server-side calculation and business rules engine.

Think .NET development is more productive than J2EE?

Think **again**.


Due to delivery pressures on our last project,
we thought about moving to .NET.
I suggested we stick with Java, but use ThinkCAP.

Think **better**.

ThinkCAP™

ClearNova's ThinkCAP is a comprehensive application platform that simplifies and accelerates the development and maintenance of J2EE-based business applications by 50 to 70%.

ThinkCAP's visual & intuitive designers bring high productivity to business developers (those with VB or PowerBuilder-like skills), content owners, and administrators while allowing J2EE architects & programmers to leverage its component infrastructure and build business logic using the tools and approaches they prefer. ThinkCAP utilizes existing infrastructure, web services, legacy systems, and business applications.

ThinkCAP saves organizations time and money—and lowers project risks. Applications are written faster and require less maintenance. Project teams utilize in-house skills and require less training. Existing infrastructure and application servers are leveraged.

With ThinkCAP you can build quality applications faster.

Learn more about ThinkCAP at www.clearnova.com/thinkcap

CLEARNOVA
APPLICATION DEVELOPMENT • SIMPLIFIED • ACCELERATED

Highly Visual Development Environment

MVC Framework with Page Flow & Actions

Advanced Data Aware Controls:
Forms, DataViews, Queries, Navigations
Workflows, Graphs, Treeviews, Grids, Tabs

Smart Data Binding™ to data, objects, XML, sessions, or requests

Browser & server-side validation

Visual unit testing with RapidTest™

Service Flow Designer aggregates Web Services, EJBs, XML, and POJOs

Content Management engine & tools

Supports .NET clients

Integrated, seamless security

Use any app server or 3rd party tool

MAY 2004 VOLUME:9 ISSUE:5

JDJ contents

Cover Story: *JDJ* Exclusive

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

Who Is Right About Java?

Who is right about Java? Is it software executive Tod Nielsen, whose advice – in reference to J2EE – is "simplify and accelerate"? Is it Eric S. Raymond, who says, as often and as loudly as possible, "Let Java go" – i.e., open source it? Is it Javalobby's founder Rick Ross, who says, "Let's rally the industry into action and create a cooperative industry alliance for Java platform marketing"? Or is it marketing consultant Joshua Greenbaum, who in an open letter to Scott McNealy recently wrote: "It's time [for Sun] to get real about enterprise software"?

Can there ever have been another technology in the past 50 years where so many people who didn't own it had such strong views on it? And herein, of course, lies the problem: it would make no sense to have a view on Java at all if it were just another proprietary technology. It is precisely because Java is positioned at the interface of closed and open technologies that we hear so many pundits offering their two cents. Everyone feels that they own Java.

Who Is Right About Java?

Asked about Java on the Microsoft platform just over a year ago, Scott McNealy's standard response was "We want the ability to interoperate and the ability for Java to have a chance to play on the desktop and on the MS server platform." Now, with the Microsoft-Sun 10-year friendship pact in place, it remains to be seen what the interoperability story will be. Cedric Beust, a senior software developer, wrote recently in his personal blog that he wants to be able to write Java code for the .NET platform.

"I don't mean writing Java on a Windows platform (I do that every day and it's working very well)," Beust wrote, "but being able to access the native Win32/.NET APIs from Java."

"Java Everywhere" takes us back to 1998, when electronics and networked computing had converged to such an extent that McNealy delivered his first ever keynote at the annual Consumer Electronics Show in Las Vegas, fired up by the prospect of Java becoming the enabling technology for everything from smart cards to smart automobiles, from set-top boxes to jewelry that could provide access to buildings.

That was the year of PersonalJava, Sun's software platform created specifically for network-connectable consumer devices; its reference implementation was announced in Las Vegas that day. PersonalJava, McNealy was certain, had the potential to generate a wealth of applications for set-top boxes by allowing developers to write software without being tied to a single operating system. From this early attempt at a smaller footprint Java for mobile devices sprang the seeds of Java on wireless handsets on a massive scale. And by February of this year Sun was announcing that Java could now be found on "250 million mobile phones, 650 million desktops, 500 million SIM and smart cards, and a hundred million other locations."

Whether it is everywhere or not, and no matter how pervasive, McNealy's Java is clearly not Raymond's Java, nor is it Ross's.

But it might well be on its way to becoming Beust's Java and Greenbaum's Java. As recently as last year, McNealy was saying – at JavaOne – "Our belief is you don't make money owning the language. You make money doing things in the language. The more people using Java, the bigger the total available market we have." What will he be saying at JaveOne this year, at the end of June? Especially now that he's made it crystal clear, as he did at the FOSE Conference in March, that Java isn't about to be open sourced any time soon. Will Sun confirm that it is indeed going to embark on a spending spree, as Greenbaum recommends, acquiring some of the Java solutions that everyone else is "building on Sun's dime"?

The best advice, it has often been said since, is to ignore advice. Life is too short to be distracted by the opinions of others. Or, as George Burns once said, "It's too bad that all the people who really know how to run the country are busy driving taxi cabs and cutting hair."

As for Scott McNealy – or, for that matter, Jonathan Schwartz, now Sun's president and COO and manifestly bursting with ideas and initiatives – I can't help thinking that they must both be wishing that everyone followed the lead of Harry S. Truman when offering sage counsel to his children. "I have found the best way to give advice to your children is to find out what they want – and then advise them to do it."

If only things in the Java ecosystem were that simple.

Jeremy Geelan is group publisher of SYS-CON Media, and is responsible for the development of new titles and technology portals for the firm. He regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas.

jeremy@sys-con.com

Java Developer's Journal

International Advisory Board

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

Editorial

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

Production

Production Consultant:	Jim Morgan
Associate Art Director:	Tami Beatty
Associate Editors:	Jamie Matusow
	Gail Schultz
	Jean Cassidy
	Jennifer Van Winckel
Assistant Editor:	Kelly Flynn
Online Editor:	Lin Goetz
Research Editor:	Bahadir Karuv, PhD

Writers in This Issue

Jerason Banes, Jason Bell, Todd Brett, Ryan Cuprak, Greg DeMelo, Jeremy Geelan, Shane Isbell, Pramod Jain, Onno Kluyt, Satya Komatineni, Stuart Lawrence, Joseph Ottinger, Kirk Pepperdine, J. Steven Perry, Gil Salu, James Scott, Joe Winchester

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

Subscriptions

For subscriptions and requests for bulk orders, please send your letters to Subscription Department subscribe@sys-con.com. Cover Price: $5.99/issue. Domestic: $69.99/yr. (12 Issues) Canada/Mexico: $99.99/yr. Overseas: $99.99/yr. (U.S. Banks or Money Orders) Back Issues: $10/ea. International $15/ea.

Editorial Offices

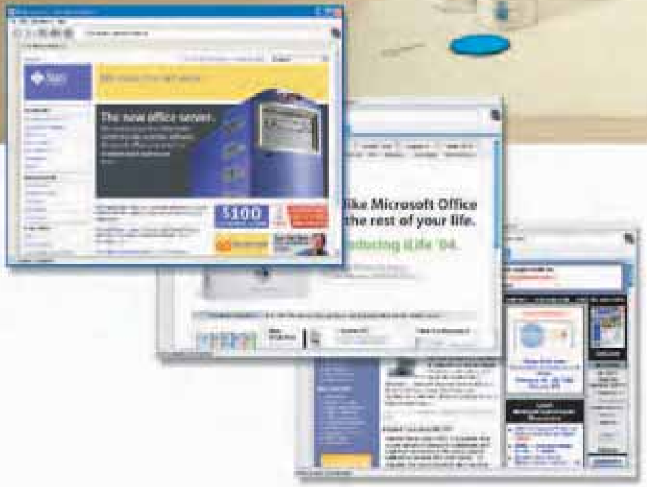
SYS-CON Media, 135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9638

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@edithroman.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

WebRenderer™



Standards compliant embeddable web browser component

WebRenderer is a cutting edge embeddable Java™ web content rendering component that provides Java applications and applets with a fast, standards compliant HTML and multimedia rendering engine. WebRenderer provides a feature-packed API including complete browser control, a full array of events, JavaScript interface, DOM access, document history and more.

Why WebRenderer?

- Standards Support (HTML 4.01, CSS 1 & 2, SSL, JavaScript, XSL, XSLT etc.)
- Exceptionally Fast Rendering
- Predictable Rendering
- Scalability (deploy in Applications or Applets)
- Security (based on industry standard components)
- Stability and Robustness

Embed WebRenderer to provide your Java' application with standards compliant web content rendering support.

To download a 30 day trial of WebRenderer visit
www.webrenderer.com

JadeLiquid™ Software

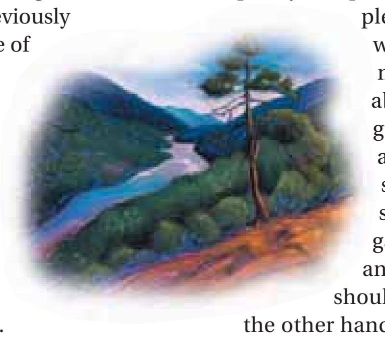
Joe Ottinger
Editor-in-Chief

Just Around the Riverbend

Two conversations over the past few days started a train of thought about where Java is right now, as did the settlement between Microsoft and Sun, the new JCP revision, and the new 1.5 JDK.

One conversation was with the author of a messaging system, talking about the use of his SDK to create a simple grid or service-based system. In short, what we talked about was a lightweight replacement for UDDI, with the conversation points being that it's far easier to manage on many levels, among other things.

Another conversation was with someone who was looking into some Java APIs that he'd previously skimped on. His sense of revelation was amusing on some levels, and gratifying on others.



I think the Java development community is holding its breath, figuratively speaking, waiting for something to happen. We're on the cusp of something, and I'm waiting to see if it's heaven or hell.

It could be either one, after all. Java's been the target of many products and vendors who see its weaknesses – often the result of choices made to create very particular strengths – as something to repair at the cost of its capabilities. Java's ability to run anywhere consistently is a strength, after all, unless you desire spectacular performance in a specific venue – but if that's what you want, and you don't care about the other platforms, then the strength becomes a weakness.

Likewise, we're seeing a lot of fairly new APIs maturing, and people are still figuring out how to use them. In this, the JCP is doing a great job, although I'm still waiting for information on normative use to be propagated. For example, I'm enjoying investigating JMX – which isn't really a new API, having existed since 1998 and finalized in late 2002 – but I have yet to see its critical mass or how people are really using it in the field. JMX isn't unique in this: the list of APIs with similar problems would include Swing, EJB, JNDI, the classloader APIs, the SPI mechanism in Java…a seemingly endless list of core concepts that people are left to fumble along with on their own power, dependent on their willingness to investigate.

I suppose it's a testament to the simplicity and power of Java that people would prefer warping its deployment rather than abandoning the language and concept altogether. That said, it gets tiresome defending the gates against people and companies who should know better. On the other hand, I'm also left wondering about the stewardship of Java. Open source advocates are screaming that it should be open, but they're also the ones who most often abrogate the core strengths of Java; Sun itself has managed to create the situation where stewardship and education are in question.

I can't say I know what's next for Java. I can tell you that every day, despite being fairly jaded, I'm impressed by the number of things you can do well in this environment, and that I find my knowledge is constantly dwarfed by the possibilities. It's frustrating, really – because some day I'd love to have a problem presented to me where all I have to do is solve it, as opposed to having to research how I'm supposed to solve it, and finding a maze of options.

What do you think?

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

josephottinger@sys-con.com

SYS-CON MEDIA

President and CEO:
Fuat Kircaali *fuat@sys-con.com*
Vice President, Business Development:
Grisha Davida *grisha@sys-con.com*
Group Publisher:
Jeremy Geelan *jeremy@sys-con.com*

Advertising
Senior Vice President, Sales and Marketing:
Carmen Gonzalez *carmen@sys-con.com*
Vice President, Sales and Marketing:
Miles Silverman *miles@sys-con.com*
Advertising Sales Director:
Robyn Forma *roybn@sys-con.com*
Director, Sales and Marketing:
Megan Ring *megan@sys-con.com*
Associate Sales Managers:
Kristin Kuhnle *kristin@sys-con.com*
Beth Jones *beth@sys-con.com*

Editorial
Executive Editor:
Nancy Valentine *nancy@sys-con.com*
Associate Editors:
Jamie Matusow *jamie@sys-con.com*
Gail Schultz *gail@sys-con.com*
Jean Cassidy *jean@sys-con.com*
Jennifer Van Winckel *jennifer@sys-con.com*
Online Editor:
Lin Goetz *lin@sys-con.com*

Production
Production Consultant:
Jim Morgan *jim@sys-con.com*
Lead Designer:
Tami Beatty *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*

Web Services
Vice President, Information Systems:
Robert Diamond *robert@sys-con.com*
Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Christopher Croce *chris@sys-con.com*

Accounting
Financial Analyst:
Joan LaRose *joan@sys-con.com*
Accounts Receivable:
Charlotte Lopez *charlotte@sys-con.com*
Accounts Payable:
Betty White *betty@sys-con.com*

SYS-CON Events
President, SYS-CON Events:
Grisha Davida *grisha@sys-con.com*
Conference Manager:
Lin Goetz *lin@sys-con.com*

Customer Relations
Circulation Service Coordinators:
Shelia Dickerson *shelia@sys-con.com*
Edna Earle Russell *edna@sys-con.com*
Linda Lipton *linda@sys-con.com*

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

Borland®
Excellence Endures™

Kirk Pepperdine
Java Enterprise Editor

All's Quiet on the West Coast

It was April 2, when I first heard the news that Sun and Microsoft had reached a settlement on their long-standing dispute over Java. When I first saw the headline, I honestly thought it was a leftover April Fool's joke, so I ignored it. It was only when I saw the words "Microsoft and Sun settle" in the CNN news banner running across the bottom of my TV screen that I came to realize that this was not a joke! It was only then that I started to think about the implications of having these two belligerents settle their differences. The question that quickly came to mind was: What does this mean for Java?

The cynical side of me wants to believe that Microsoft needed to settle, after all they now have to appeal huge fines and penalties that have been levied against them by the European Commission. Just as they needed Apple in the past, they need Sun now. A sign of healthy competition can only help Microsoft in its own ongoing "anti-trust" battles. Even though a real settlement with the European Commission is still most likely five years away (due to the appeals process), the recent ruling made by the EC would appear to come close enough to Microsoft's settlement with Sun to have real meaning. In fact, it's been clear for many months that the EC was going to rule against Microsoft. The remedy imposed by the courts includes an order for Microsoft to freely provide the information about Windows to others.

The next question is, why would Sun decide to settle? Unlike the government's anti-trust case against Microsoft, Sun had some very compelling evidence. From that aspect it seems reasonable that Sun should win and they did. Though a settlement may have denied Sun their day in court, it also means that they got what they needed from Microsoft – that Microsoft will now bundle updated versions of Java on all of their Windows products. The lesson here is that if you can come to an agreement on your own, why have one imposed upon you?

> "The lesson here is that if you can come to an agreement on your own, why have one imposed upon you?"

In the end, what I (as a developer) want to know is what the settlement means for Java. If you look deeper into the settlement, you'll find that Sun has also agreed to license the Windows desktop system communication protocol. Sun has been trying forever to position Java on the desktop; does this mean they will finally be able to get there? Will the use of the Windows communication protocols break Java's WORA guarantee? Not according to a recent blog by James Gosling. In that blog, James goes on to explain that although they can use the technology, it doesn't mean that they will be locked into Windows. So again, this looks like a real win for Sun.

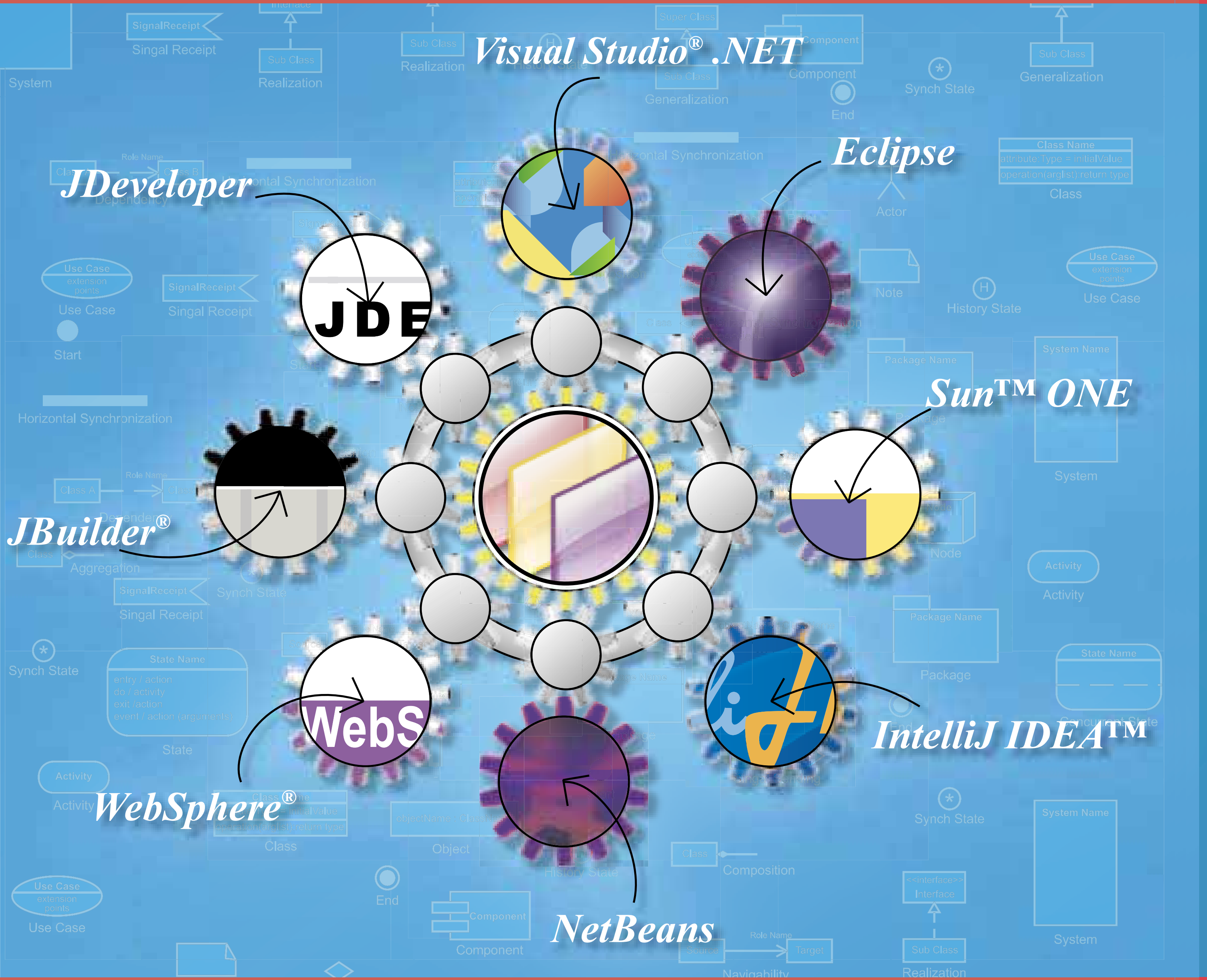
What we have is a mature market that has a strong preference for bundling and, since Microsoft owns the platform, it gets the final word in what goes into that bundle. Although this solves the apparent problem of the distribution of the Java runtime for Sun, does it solve the real problem? Those of us who are distributing Java applications to the desktop know what the answer is: not really. The reason being that just as a Windows application (like the Java VM) is dependent upon having very specific services and versions of the core DLLs available to it, a Java application often requires a very specific version of the Java runtime. As a result, deployment teams are still going to need to distribute the appropriate runtime with their application unless they are willing to spend the time testing it against all possible versions of Java.

There is no doubt that there were a multitude of reasons for Sun and Microsoft to settle, many of which we will never hear. It certainly would be interesting to know what effect (if any) the recent EC ruling had on bringing Microsoft to the table. It would also be interesting to see if this ruling helps Microsoft with the EC ruling. That said, dispensing with the legal distraction can only be a good thing for it will free up a lot of people's time to focus on Java. Now maybe Sun can show the world how to turn lawyers into Java programmers.

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

kirk@javaperformancetuning.com

Model Once, Code Anywhere



Build Quality Applications Faster, Better and Cheaper

VisualParadigm *for* **UML**

software development 14th annual productivity award

VisualParadigm **SDE**

Visual Paradigm for UML was awarded SD Magazine's 14th Annual Jolt Productivity Award (along with Borland® Together® for Eclipse) in the Design and Analysis Tools category, over IBM® Rational® XDE.

Powerful features include:

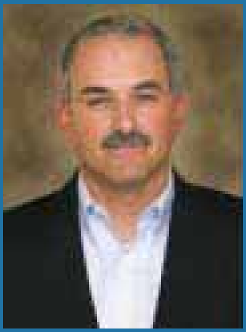
- Latest UML notation support
- Java, .NET, XML, C++ IDL reverse engineering
- Real-Time Code-Model synchronization
- Full software development process support
- Seamless integration with Visio *NEW*
- Teamwork infrastructure *NEW*
- OLE support
- Intuitive modeling interface
- Report generation
- Plug-in open architecture

Visual Paradigm

www.visual-paradigm.com
sales@visual-paradigm.com

Interview by Jeremy Geelan

– THE – FUTURE OF MIDDLEWARE & TOOLS


Robert LeBlanc
General Manager
Tivoli Software

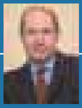
John A. Swainson
General Manager
Application & Integration Middleware Division

Janet Perna
General Manager
Data Management Solutions

Ambuj Goyal
General Manager
Lotus Software

Mike Devlin
General Manager
IBM Rational

Exclusive Q&A with the 5 general managers of IBM's Software Group



John Swainson, WebSphere

Q Let's start at 35,000 feet: What's been the main impact of IBM's "On Demand" vision on the Information Technology scene to date?

Swainson: Let's start by defining "on demand." First, on demand reflects what our customers are doing with their businesses – streamlining their business processes to make them more flexible and adaptive to new markets and opportunities. They use information technology as a tool to integrate these processes, so obviously IT is a critical enabler of on demand. In the case of IBM software, our focus has been on making sure customers can use our products – WebSphere, DB2, Lotus, Tivoli, and Rational – to establish what we call an on demand operating environment, which is a middleware infrastructure that gives our customers a way to build, integrate, and manage their new and existing applications.

Making it possible for people to do this required that we create a new, standards-based computing model. This model is based on the technical principles of a service-oriented architecture. It is delivered in our products, ones like WebSphere Business Integration, that customers can buy today. However, more and more, we are seeing customers starting to approach this from the business process angle, using our business consultants to define the business processes, and then tying the delivery of those processes to a set of technology models that help them achieve their goals.

Many customers have begun to realize the business benefits of on demand. Charles Schwab has implemented a WebSphere-based solution to allow their financial advisors to deliver portfolio analyses to their clients in near real time. By taking advantage of grid computing features in WebSphere, they have been able to make more efficient use of server resources and reduce the elapsed time to run these transactions from 8 – 10 minutes to just 15 seconds.

Q How favorably does it resonate with your customers, the fourfold typology of 21st century businesses as needing to be responsive, resilient, variable, and focused? Are there any further attributes that have emerged as being equally important?

Swainson: On demand resonates very well with customers because it maps to what businesses today are trying to do – find ways to be more flexible and responsive in order to stay competitive. It manifests itself in things like our WebSphere and Tivoli products, where systems become self-monitoring and self-healing, cutting down on system outages and enabling employees to spend more time on higher-level tasks. We're also seeing more interest in business process integration projects geared to specific industries.

Q How large do SOAs loom in IBM's vision for the future?

Swainson: IBM has been helping customers build service-oriented architectures for more than 10 years, because SOA is a design pattern, not a product per se. We have thousands of customers who have built these on WebSphere and MQ. Based on this experience, we have worked with the industry to define a set of standards and have codified successful implementations into a set of templates and guidelines for our services teams to use.

We have worked with thousands of customers on Web services and SOA engagements, and we lead the industry in every aspect of SOA, including products, standards, education, services, and experience. Our work includes standards created by the big market share of products like WebSphere MQ enterprise messaging, as well as more formal standards. IBM was a primary driver behind the Web Services Interoperability Organization, and we work with other vendors to introduce specifications for transactions, reliability, management, and grid computing. JSR109 is a recent example.

Q And grid computing – where does that fit in?

Swainson: Grid computing is another technical underpinning of an on demand environment. We've been helping our customers develop grids for years, primarily for scientific analysis. Now IBM is introducing technology that can be applied to commercial grid environments to make them easier and less expensive to implement.

Q In terms of the competitive landscape, since Gartner reported that 37% of all deployed application servers were WebLogic vs. IBM (at 22%), things have changed dramatically. What are the current statistics in terms of market share between you and BEA?

Swainson: Last year Gartner reported that IBM's share of the application server market rose to 37% in 2002 from 31% in 2001, making us the market leader. BEA dropped to second place, with market share declining 5 points, to 29%. Our goal was to become the market leader, and we think we're extending our lead.

Q When talking to your customers, what are the key integration drivers from their perspective, and how is IBM delivering on each of them?

Swainson: Integration is a must, but the integration priorities of customers are unique depending on the size of the business, the industry they compete in, the customer's long-term goals, and the customer's technology capability. There are a couple of key themes that rise to the top. Customers want to link different areas of their business with those of their clients and partners, and they want to leverage as much of their existing infrastructure as possible.

We help them get there with IBM's open, standards-based approach to integration, with WebSphere. We enable customers to automate the flow of information and process transactions to help them become more agile and responsive. We use horizontal end-to-end integration to leverage existing investments to increase business flexibility and efficiency and drive ROI. IBM can tailor integration solutions to best suit the needs and priorities of any customer, no matter the industry. IBM offers 63 industry-tailored software solutions spanning 12 industries. And every one of those 63 solutions is based on WebSphere.

Q How about the breakdown between large enterprises and SMBs – where does IBM see the most growth? What kind of things are you doing for the SMB space?

Swainson: The SMB marketplace represents a big opportunity for top line revenue growth for IBM. According to AMI-Partners, the overall market opportunity for SMBs last year was $300 billion, and it is the fastest-growing segment of the IT market – growing faster than the IT market as a whole. SMB is also the fastest-growing market segment for IBM. We just announced a strong start to this year with first quarter SMB growth at 15% year to year – thanks in large part to a partner ecosystem that delivers more than half of IBM's total SMB revenue. Local and regional ISVs represent a cornerstone of IBM's SMB strategy – more than half of all mid-sized customer IT investments are based on the application decision.

Also, while SMBs may not have the size, global reach, and revenues of larger organizations, they still have the desire to have a successful, profitable, and growing business. IBM's Express portfolio of middleware, hardware, services, and financing is a comprehensive suite of offerings built from the ground up for SMB customers and priced with their needs in mind.

Q Jonathan Schwartz went on record in *JDJ* as saying "middleware is history." Clearly that wasn't meant literally, but he was saying that end-to-end "systems" will supplant it as a focus. Is the IBM view that middleware, on the contrary, is just beginning?

Swainson: Saying that middleware is "history" is laughable. IBM has tens of thousands of customers who need and use middleware for transactions, data management, development tools, systems management, security, and collaboration in a heterogeneous systems environment. The WebSphere platform experienced 12% revenue growth in 2003 over the previous year. The WebSphere platform grew 24% in 1Q04, marking its twenty-second consecutive quarter of revenue growth.

Q Scott McNealy once said "We're down to three – IBM, Microsoft, and Sun." Does the recent Microsoft-Sun pact change anything as far as IBM is concerned?

Swainson: Sun is a distant fourth or fifth in middleware market share, depending on which study you're looking at. We haven't heard much about what the deal really means for Sun, Microsoft, and the future of Java, but we haven't heard that Sun is backing away from Java either. I expect that Microsoft will continue to try to convince people to move from Java to their proprietary language environment, and I expect that Sun will continue to strongly support Java.

Q Some of IBM's moves in the Java space appear to be moving away from Sun – Eclipse tooling rather than NetBeans, SWT instead of Swing, and now a new virtual machine. From one perspective it looks like IBM is trying to break away from having any licensed Sun code in its product offerings. Is this the end game?

Swainson: IBM created Eclipse because there was no industry standard for building and integrating application development tools. We see Eclipse as very supportive of Java, and it has made a large contribution to Java's success in the marketplace – and with more than 18 million downloads, it is by far the most widely adopted Java development environment. Interestingly, it is also being broadly used in non-Java environments as well, with a great deal of activity in the community now to build C and C++ tools based on Eclipse.

SWT was developed because many of our customers demanded user experiences that the Java Swing code couldn't provide. The whole notion behind Swing is to have a consistent UI model across clients; SWT, on the other hand, allows people to build UIs that are deeply integrated with the look and feel of the host platform. So you can build Windows applications with that look and feel that fully integrate into the Windows desktop, and the same goes with Linux and pervasive platforms. Having said that, Swing is part of the Java standard, and IBM delivers more products using Swing technology than anyone else in the industry.

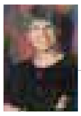
Q Scott McNealy told IBM to stop pushing him to open source Java until you do the same with DB2. If Java is not open sourced, how does this affect IBM's licenses with Sun in the future? How serious was the call to Sun to collaborate with you on an open source implementation of Java?

Swainson: We have been stating for years that we'd like Sun to make Java a true open standard, and we remain optimistic. IBM's long-standing support for open source is based on our conviction that openness creates new opportunities and spurs innovation. Open source also gives customers choice and helps them meet their IT needs more quickly and effectively. An open source Java platform would be good for the industry, good for customers, and good for Java.

Q What major product announcements is IBM likely to make this summer, or should we be waiting for the fall for the next big additions?

Swainson: Later this year we'll be announcing the next generation of WebSphere Application Server, Version 6. It will be the foundation for IBM's on demand operating environment and will advance IBM's leadership in things like grid and autonomic computing, security, systems management, application development, and SOAs. We'll also begin rolling out software that will closely integrate IBM's market-leading WebSphere MQ messaging software with high-level integration to form a single Enterprise Service Bus infrastructure.

All these products are part of IBM's effort to help customers compete more effectively in the marketplace. We will continue to help customers solve pain points, such as integrating far-flung silos of information across the enterprise and with trading partners. And we will continue to deliver on IBM's on demand vision through industry leadership in open standards, ongoing investment in research and software development, strategic acquisitions of companies like Rational and, more recently, Trigo and Candle, and our unmatched portfolio of software, hardware, services and financing. That's how we are making on demand a reality for our customers.



Janet Perna, DB2

Q How does DB2 rate at the moment in terms of the competitive landscape, versus Oracle for example?

Perna: For 2004 and into the future, our goal is to continue to be the best at providing exceptional customer value. This means offering the best technology, the best solutions, the best service and support, and incomparable time-to-value for our customers.

Our information management efforts are focused on helping customers manage their growing requirements, enabling them to more easily integrate, manage, and gain value from their business information. Companies today are not only challenged with finding ways to integrate and manage hundreds of different database systems that are scattered throughout their organizations, they are also struggling to manage information based on a wide variety of formats, across different vendor platforms and in silos throughout their enterprise. They want to be able to leverage the information within these systems to improve their operational efficiency, customer relationships, and productivity of their people.

IBM provides customers with the broadest capabilities to meet these requirements, allowing them to streamline business processes, gain deeper insight into their business information, and ultimately gain a competitive edge in the marketplace.

Q What impact does your division have on IBM's on demand strategy? How are you delivering on this?

Perna: IBM's on demand strategy is about enabling enterprises to respond with speed to any customer demand, market opportunity, or external threat by integrating business processes across the enterprise and with key partners, suppliers, and customers. Our information management technology is at the heart of this effort, helping customers build a flexible and responsive information infrastructure that enables them to respond dynamically to changes in their environment, industry, or economy.

Most enterprises have silos of applications and information, which have evolved over the years. Now they want to automate these systems across the enterprise and horizontally integrate them so that they work in a “demand to deliver” environment. Our unique offerings help customers automate the flow of information throughout the enterprise and enable them to analyze it in real-time, placing business information at the core of their business strategies. As a result, they can streamline business processes, improve inventory flows, and boost customer service. And of course, they can protect their existing IT investments. These are all key attributes of our on demand strategy.

Q What are the key factors driving the information management marketplace?

Perna: Clearly, there is a dramatic shift occurring in the marketplace to help companies drive more value from their business information. Key factors driving this market include the ongoing data deluge; emerging technologies such as pervasive computing and RFID, which are generating even more amounts of data; and the increasing number of people that need access this information. Additionally, there are also new requirements being placed on companies through compliance mandates. Information such as documents, e-mails, and instant messages needs to now be retained, managed, accessed, secured, and searched to bring business value, and help build a technology foundation for compliance.

Q Why is enterprise content management so crucial for companies and what is IBM's strategy in the marketplace?

Perna: As the amount of information continues to grow, it's also being developed in unstructured formats such as e-mail, video/audio, and business documents. In fact, today, more than 85% of business information is being developed in unstructured formats. Customers are looking for ways to harness this information, gain a consolidated view of it, and link it to core business processes. Furthermore, new regulatory compliance mandates like Sarbanes-Oxley and HIPAA are calling for increased scrutiny of information stored in e-mail and other documents, making an integrated content management solution an essential part of a company’s IT infrastructure.

Content management continues to be a major focus for IBM this year. In addition to targeting small- to medium-sized businesses, addressing vertical markets and rolling out industry-specific middleware solutions, we have also upped our investment in research and development, increased our CM sales staff, and made three acquisitions for CM technologies in the past two years. No other vendor can provide the breadth of capabilities that IBM delivers today.

Q What are you doing to better support Java?

Perna: Since partnering with ISVs and application developers is strategic for DB2 and IBM's middleware business, we have a rich tradition of supporting developers by providing them with software and tools that are easy to use and easy to develop applications on. DB2 provides the broadest support for open standards, and this approach continues to give DB2 customers and developers maximum flexibility to develop database-driven applications in any programming environment. For example, the next release of DB2 will enable programmers to use the popular Eclipse toolkit to more easily and cost-effectively develop applications that work with DB2. Likewise, the next release of DB2 will take advantage of new .NET development tools well before SQL Server users can.

Q What are you doing to appeal to the developers that cater to the mid-market?

Perna: Last year we delivered our DB2 Express and Content Manager Express offerings targeted at mid-sized customers. We have seen strong support from developers for our express portfolio due to its ease of installation and platform flexibility. Specifically, DB2 Express eases application development by delivering numerous tools for automating and simplifying database functions. DB2 Express offers developers maximum flexibility, supporting XML, Web services, and Java, while simplifying development of .NET applications through the delivery of new tools that integrate seamlessly into Visual Studio .NET. Content Manager Express has helped us extend the reach of our CM offerings to the SMB segment, offering smaller organizations and departments a tool for managing digital content such as documents and image files.

Q How great a role does IBM see for Linux in combination with DB2?

Perna: Linux has always been a strategic platform for our information management portfolio and will continue to be as a growing number of our customers are standardizing on it to run their enterprise applications. IBM was the first major IT company to embrace Linux, enabling our middleware and hardware to run on it for early adopters. For example, DB2 provides the broadest support for Linux in the industry, from handhelds to the mainframe to clusters of mainframes. DB2 was the first database to support Linux, the first to support clustered Linux servers and the first to support Linux for the mainframe.

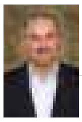
Last year we introduced, DB2 ICE, an integrated DB2 cluster offering for Linux on our xSeries eServer platform. DB2 ICES scales much higher and deploys faster than anything else available in the marketplace. This year we are extending our database leadership on Linux with specific enhancements to the next version of DB2, that will help database clusters scale higher and perform faster. The new offering will also support the new Linux kernel (Version 2.6) that better exploits DB2's 64-bit capabilities and takes better advantage of servers that use multiple processors.

We also see strong demand for DB2 on Linux in the SMB space with DB2 Express. Nearly 65 percent of solution developers who will use DB2 Express tell us they prefer the Linux platform.

Q What sets you apart from the competition, such as Oracle?

Perna: We are the only vendor focused on customers’ complete information management requirements – content management, information integration, database management, and business intelligence. One of the unique approaches that set us apart from the competition is the ability to help customers integrate, access, and gain intelligence on data sources from a variety of vendors. While our competitors promote a vendor lock-in, or rip and replace approach, IBM will help companies extend their IT investments by enabling them to leverage their information assets wherever they are.

An on demand business is an information-based business. IBM is providing the framework for the evolution of the information infrastructure, which will be required to deliver information on demand. No other software company is taking this approach to information management.



Robert LeBlanc, Tivoli

Q IBM invested in Tivoli years ago for its systems management framework. Tivoli struggled in recent years – what changes took place to strengthen the business, and to integrate with the rest of IBM's software offerings?

LeBlanc: Since the acquisition of Tivoli by IBM, there have been a number of events and efforts that have helped shape the brand into what it is today.

Features, Performance and Control

Discover the ILOG JViews Graphics Components

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

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

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

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

ILOG®

Changing the rules of business™

When I took over as GM in mid-2001, I spent the first 60 days visiting 45 customers. We were right on target with our open standards and cross-platform approach, but had to fine-tune the delivery, so we made changes based on what we learned from these meetings. Essentially, we learned we were confusing customers with our offerings and we weren't taking full advantage of the technology in other brands. We also learned that our customers were looking to reduce the cost of operating their IT systems while maintaining growth. So, we focused on a few key areas and modified our strategy.

First, we grouped the business into a few key segments – core systems management (performance, availability, configuration, operations), security, and storage – while simplifying the product offering from more than 150 products to approximately 50.

Next, we began to leverage key IBM technology, starting with WebSphere, shifting to a modular approach for product delivery. Customers have unique needs. One may need the complete portfolio, another might want to add an identity management component to limit security risks, while a third could require storage software to meet compliance regulations. By using WebSphere, and J2EE, as the underpinning technology, customers had easier to implement, easier to deploy management solutions designed to fit their specific business needs.

Following these steps, we recognized that the broader challenge facing customers was the need to shift away from the costly, inefficient "reactive" method of systems management, and move towards a "proactive," predictive model. So we delivered technology solutions built around business policies and needs. This means that corrective responses to security threats or server crashes are now automated, freeing up resources (people, technology) to focus on the more pressing problems of the business.

It's important, as well, to note that our business has grown organically though acquisition and by partnering with industry and market leaders. We have strengthened our portfolio through innovation – continuing to develop our existing product portfolio and driving the IBM on demand initiative. Acquisitions have bolstered all of our segments; security and storage through Access360 and TrelliSoft, respectively; and the on demand management space with industry-leading technology from Think Dynamics. Finally, partnerships with the likes of Cisco, Citrix, and Siebel help us deliver industry-specific solutions to the broadest set of customers.

This focus on the customers, on meeting our customers' unique needs, is having a positive effect on our business: in the first quarter of 2004 we grew 18%, our fourth consecutive quarter of growth (on top of 12% growth for 2003).

Q **Where would you position IBM with respect to CA, Veritas, EMC, and others?**

LeBlanc: The way in which I perceive us versus the companies you mention is largely based on our heritage and focus. IBM has a broader view of systems management than any other vendor; we provide customers with storage, security, and network management and we integrate with a larger portfolio of products and services that make our offering more cohesive. It's about flexibility and choice. Add to that our integration with the other brands – particularly DB2 and WebSphere – and we have a set of solutions our competitors cannot touch.

Our competitors approach systems management in different ways. Some are focused on specific areas, such as storage, which limits their scope and their ability to look at the entire IT environment. Some of our competitors are not software companies by trade and therefore are in the process of quickly acquiring technologies and learning about the software industry to catch up to the market. Some are reconciling with their customer base while trying to figure out what their role will be in an on demand world.

Our strategy is well based on a solid foundation that revolves around customers needs. We approach systems management as a critical element of a larger ecosystem that we need to keep healthy, lean, and efficient.

Q **Tivoli Storage Management Software recently became part of IBM's TotalStorage offering. What drove the decision to join the sales teams? Will the Tivoli brand remain or be assimilated into the TotalStorage offering?**

LeBlanc: The decision to merge IBM Tivoli storage management software and IBM TotalStorage sales teams was a decision driven by the needs of our customers. Storage management software has become an intrinsic element of any storage strategy. Our research indicated that customers prefer to buy complete integrated storage solutions. We had all the pieces of the puzzle, so it made sense to integrate our sales efforts.

Furthermore, by re-branding the IBM Tivoli storage products under IBM TotalStorage Open Software Family, IBM has been able to strengthen its storage software position by bringing together two key elements of the on demand environment – automated storage management and storage virtualization software. We are now better able to deliver a "TotalStorage" solution that includes end-to-end storage hardware and software while continuing to deliver software that optimizes customers' heterogeneous environments.

Q **What is Tivoli's role in the on demand initiative, specifically around automation; why is automation important to customers?**

LeBlanc: Tivoli's portfolio and technology encapsulates the essential elements of automation – provisioning, orchestration, availability, security, and optimization – that are critical elements of an on demand environment.

So why is it important? Quite simply, our customers are dealing with increasingly complex systems, limited budgets, and dynamic business environments to which they need to respond quickly. We are providing customers with the tools to better respond to market conditions, competitor moves, and regulatory requirements. Their IT infrastructure needs to be designed to proactively keep pace with these changing conditions and put recent and relevant industry trends in the proper business context.

Automation is important because it will reduce the costs and manual efforts needed to maintain and manage complex environments.

Q **IBM has been touting identity management as a critical element of on demand. How are you making it easier for developers to integrate identity management into the applications that make up an on demand environment?**

LeBlanc: Before delving into how IBM has helped developers meld identity management technologies into the on demand environment, I think it's important to first define IBM's view of identity management, which goes beyond traditional user provisioning. From our point of view, access management and privacy management are critical to any comprehensive identity management solution. IBM has made strides in each area to ensure developers are provided with high-integrity, flexible security management offerings that are also easy to implement and integrate into any environment, on demand or otherwise.

This is evident through IBM's role in developing Web services security standards initiatives and our support for bodies such as WS-Security and OASIS. IBM is a leading proponent of Web services and has assisted in the development of widely adopted security standards like SAML. IBM has also developed the first ever privacy programming language, the Enterprise Privacy Authorization Language (EPAL), designed to give developers the ability to extend specific privacy rules across internal business systems and automate compliance to those rules. In December, the specification was submitted to the W3C for continued development and eventual standardization.

In general, IBM Tivoli offerings have long been touted for their interoperability. The security management portfolio is no different – IBM Tivoli security products are Java-based and support XML and different specifications including SAML.

Taking all this into consideration, developers using IBM Tivoli to bring identity management technologies to the forefront of their IT systems are at a distinct advantage – IBM is unique in that it combines industry leadership with comprehensive identity management solutions to create a truly valuable environment for developers.

Ambuj Goyal, Lotus

Q **Lotus remains to most people associated with groupware, its prime invention. What role does Lotus play now in the technology landscape? How does it fit for example into IBM's "On Demand" vision?**

Goyal: Lotus plays a vital role within IBM's on demand initiative. By creating new work environments designed to raise organizational productivity, improve communication, and extend the reach of employees, IBM Lotus software helps organizations to become on demand businesses. Lotus represents the people side of on demand, enabling the greatest resource a company has – its employees – to adapt, act, and turn information into action.

The Lotus Workplace and Lotus Notes/Domino product families are how businesses connect their people with their business processes; they are the collaboration component of IBM's on demand strategy. IBM WebSphere Portal is how businesses provide their people with single, integrated access to the information, applications, and people they need, in the context of their role. Together, they can make organizations become more focused, responsive, and better able to compete in today's ever-changing marketplace.

Lotus Workplace enables people to interact, communicate, and collaborate across diverse computing systems in a single workplace environment. An organization realizes the value of e-business on demand when its people and processes are integrated end-to-end across the organization – and with key partners, suppliers, and customers.

Q How big a driver for Lotus is customer choice?

Goyal: Customer choice has always been important to IBM. It is why we have created a standards based model for delivering products and services across the on demand operating environment. The flexibility a customer gets with IBM software allows them to decide their future IT investments. We adapt to customers needs; we do not expect them to adapt to our technology.

To meet customer needs, Lotus is delivering flexible, portal-ready collaboration components, built using Web services and the J2EE architecture that can be easily customized to fit a specific industry or business. The modular, on demand design of Lotus Workplace allows customers to pay for what they use, which leads to reduced total cost of ownership.

At the same time, Lotus Notes and Domino can be a big part of a Workplace solution. IBM will continue to support Lotus Notes and Domino. With over 100 million users, it is proven technology that provides value to customers around the world. Investments in IBM Lotus Notes and Domino can be leveraged because they will continue to function seamlessly as an integral part of Lotus Workplace.

The Lotus Workplace collaboration platform is comprised of industry leading software from across the IBM Software Group brand teams. Lotus Domino's rapid application development capabilities will continue to be available, as well as new Workplace Builder capabilities being released in May. We will also provide new capabilities to leverage Domino applications in Lotus Workplace. New tools, for example, are being developed that will help customers bring new and/or existing Domino applications into the dynamic portal environment of Lotus Workplace.

Q How would you summarize the value proposition of Lotus Workplace?

Goyal: Lotus Workplace integrates people with business processes. We make more people more productive in the context of their work by giving them access to the information, processes, and other people they need in order to move a process forward. Lotus Workplace improves organizational productivity by giving employees an integrated set of collaborative tools that can be tailored to their specific role or work environment. Instead of dealing with several separate collaborative applications, the Lotus Workplace strategy consolidates collaborative capabilities into a single, integrated work environment. As a result, people can quickly collaborate to address customer demands and react to changing market dynamics and competitive threats. Whether a user works from a remote location, in a team environment, or on several projects at once, Lotus Workplace products provide an integrated platform that can help enhance employee productivity and enable collaboration across an entire value chain.

Additionally, Lotus offers customers value through several Workplace business value offerings, such as Lotus Workplace for Business Controls and Reporting, designed to help companies manage processes, controls, and information that may be useful in compliance with the internal-control reporting requirements of Section 404 of the Sarbanes-Oxley Act.

Q What's the reaction of ISVs to the direction that you are moving?

Goyal: Our ISVs have identified Lotus Workplace as a significant new business opportunity. Because of the open nature and easy integration of Lotus Workplace products, our partners have more time to focus on true value add and do not have to worry about integration. Pre-integration of collaborative capabilities on an open, unified infrastructure has made that possible. As a result our partners are quickly deploying, tuning, and customizing workplace solutions for their customers, accelerating their application reach and increasing revenue.

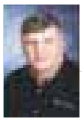
Our Lotus Notes and Domino partners are excited as well. While many of them will continue to develop applications with Lotus Domino, most realize the additional value Lotus Workplace can provide. In the end, it is the customers that win. We are providing a clear path for customers to stay where they are, or adopt the Lotus Workplace model. Either way, their investments in Lotus Domino applications will be preserved.

Now, there are more opportunities than ever for ISVs with our recently announced IBM Workplace Client Technology. With this technology, ISVs in particular will be able to extend the value of existing Web- or Java-based applications, and develops new applications that will provide users with a rich user experience. This includes many rich client functions, including disconnected use, and is still centrally managed just like the network centric applications they are deploying today.

Currently, no other software vendor is offering this type of integrated collaborative platform on which partners can build customized applications and business solutions – applications and solutions that can be deployed on a variety of clients, including rich clients, Web browsers, portals, and mobile devices. At Lotusphere this year, more than 50 business partners announced news, capitalizing on the new Lotus Workplace platform and demonstrating their commitment to working with IBM Lotus software.

Q What's happening with Lotus Freelance Graphics?

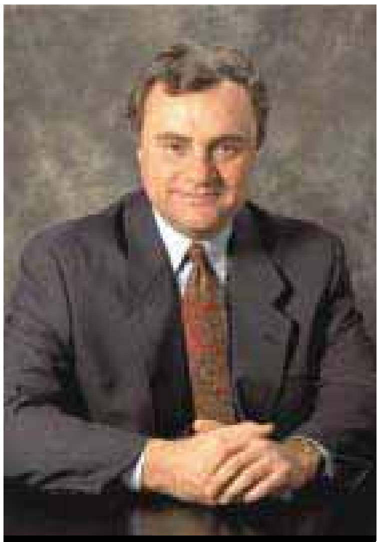
Goyal: Lotus Freelance Graphics continues to be offered as a capability in Lotus SmartSuite. SmartSuite Release 9.8 is the latest edition of the award-winning office suite including Lotus 1-2-3, Lotus Word Pro, Lotus Freelance Graphics, Lotus Approach, Lotus Organizer, Lotus FastSite, and Lotus SmartCenter.



Mike Devlin, Rational

Q It's now nearly 18 months since IBM's announcement to acquire Rational. What's been the main impact of the acquisition, so far as you are aware, on the world of software tools development?

Devlin: There are a number of areas in which our customers have and will continue to see dramatic improvements in the software development solutions that IBM provides. Many of these improvements have resulted from the deep integrations between solutions within the IBM software group.


Steven A. Mills
Senior Vice President & Group Executive
Software Group

Steve Mills was appointed Senior Vice President and Group Executive, IBM Software, in July 2000. In this capacity, he is responsible for shaping IBM's overall software strategy and directing IBM's $14 billion software business.

IBM's industry-leading middleware products power the e-business infrastructures of virtually every mid- to large-size company in the world, and IBM holds the number one or two position in marketshare in all major software markets in which it competes according to industry analysts.

Today, Mr. Mills is leading the next phase of IBM's software strategy, through which IBM is delivering industry-specific middleware solutions to customers in 12 key industries. This includes the development and marketing of new industry-specific offerings as well as aligning IBM's software salesforce (the world's largest direct sales and support team, with more than 13,000 people) along technical and industry lines. In addition, Mr. Mills is leading a series of programs for Independent Software Vendors (ISVs) to help them deliver industry-vertical applications running on IBM's middleware.

Mr. Mills has played a leading role in the growth of IBM Software Group since its inception in 1995. He was General Manager of IBM Software Group Strategy and Solutions, responsible for IBM's strategy for middleware and software solutions for e-business, as well as managing business units for Business Intelligence Solutions, Pervasive Computing, e-Commerce Solutions, and Solution Technologies

He joined IBM in 1974 as a sales trainee in New York City and was a marketing representative until 1980. In 1981, he joined the business planning staff of the Data Processing Division and became manager of that function a year later. In 1984, he was named Administrative Assistant to the IBM Vice President and Assistant Group Executive of Plans and Controls in the Information Systems Group.

He became Director of Planning in the Information Systems and Communications Group in 1985. In 1986, he was one of the executives responsible for starting IBM's Publishing Systems Business Unit. He became Director of Financial Planning at Corporate Headquarters in 1988.

He joined the Programming Systems line of business in 1989 as Programming Systems Director of Operations. He was named Assistant General Manager, Finance and Planning, for that organization in December 1990, and in December 1992 became General Manager of the division's Santa Teresa Laboratory. In 1993 he became General Manager of IBM's Software Solutions Division.

An example of this is the IBM Software Development Platform, a proven approach for business and technology leaders who recognize software development as a key to business transformation.

The IBM Software Development Platform is a comprehensive set of products and proven best practices for teams who build, extend, modernize, integrate, and deploy software. It is a cross-IBM Software Group solution that includes 18 core products and dozens of complementary and technology-specific extensions, enabling clients to choose the optimal solution for their team and technology environment.

The IBM Software Development Platform enables customers to embrace software development as a strategic business process like enterprise resource planning, customer relationship management, and human resource management. This allows customers to automate and integrate their processes across their organization, enabling companies to be more focused, responsive, and resilient, which can yield both top-line benefits and bottom-line results.

Q What's the current status of Rational's approach to software development, Rational Unified Process?

Devlin: RUP continues to lead the market as the de facto industry standard for iterative software development. In the past two years, RUP has evolved into a comprehensive but flexible process platform that allows for customization for specific customer project needs.

With the acquisitions of both Rational Software Corp. and PwC Consulting, IBM is in the fortunate position of owning two of the leading commercial software methodologies: IBM Rational Unified Process and IBM SUMMIT Ascendant. Each of the commercial methods brings significant and established customer bases. IBM also owns the IBM Global Services Method, which is used by IBM practitioners on engagements and on IBM accounts. These methodologies and offerings are quite complementary, and IBM expects customers will benefit greatly from the combined strength of IBM's full suite of methodologies.

In the short term, IBM plans to continue to support both the RUP and SUMMIT Ascendant offerings, and customers will be able to choose one or more of these methodologies, depending on their project needs and environment. For the long term, IBM plans to examine the best way to leverage the combined strengths of the IBM methods and tools – IBM Global Services Method, IBM Rational Unified Process, and IBM SUMMIT Ascendant – to lead the methodware market.

Q IBM tooling has tended to focus around its core runtimes. Rational, however, is tooled for non-IBM runtimes including .NET, Oracle, etc. What is the future of tooling in this space?

Devlin: Our customers live in a heterogeneous world. Rational will continue to support non-IBM runtimes, and wherever possible seek to advance standards that allow for maximum interoperability. IBM's work with Microsoft on Web Services, BEA on Simplified Data Objects, and the OMG with UML 2.0 are examples of this effort.

Rational will continue to develop new products that are designed to work in .NET, J2EE, and other runtime environments. One reason that we have kept the Rational brand identity is to allow for some separation between our WebSphere runtime environment and tools specifically designed for it, and more general software development tools, which are appropriate for heterogeneous development.

Q With so much functionality in Eclipse, which is open source, what's the business model that allows IBM to continue make money from its tools? Does "professional open source" resonate with IBM as a useful concept to describe what the new model is?

Devlin: Organizations will have a choice to make as they move into the next generation of software and systems development. That choice is to place their development information into a closed proprietary information model, or to use an open set of frameworks that they have complete access to, that are completely transparent in their implementation. Eclipse is about openness and competing based on value, not on lock-in.

For the last 40 years, the basic building blocks of software development tools, shells, debuggers, loggers, editing windows, meta-models, etc., have been continually reinvented. So much time has been spent on the basics that the real value to customers can get lost in the shuffle. Eclipse levels the playing field for the basic pieces that make up the development environment. It provides open components for everyone to use as the core of their solutions.

As adoption of those building blocks grows, more attention can be provided to focusing on value at the next level up. There are more than enough customer needs to build a software development business model. The next successful wave of solutions will be those that can take the basic components that are part of the Eclipse framework and put them together in ways that allow teams to go faster, to get closer to their business users, and closer to those that deploy and maintain their solutions.

Q How different is it for you personally, being one of IBM's software GMs instead of Rational's CEO? How do the day-to-day challenges differ from those you faced before?

Devlin: The past 14 months have been a very exciting time for me personally. Being part of IBM has provided Rational with access to a vast amount of engineering resources and research. This has enabled us to more tightly integrate our software development solutions with those from the other brands of the IBM Software Group. It has also empowered us to continue our long track record of innovation and thought leadership in the software development industry. The acquisition by IBM has truly enabled us to be more successful in achieving our goal of ensuring the success of our customers.

Q Is it the intention that the Software Development Platform you announced last month will be interoperable with all five brands? Is that what will unite you all more than ever, moving forward?

Devlin: Yes, the IBM Software Development Platform is about treating software development as a strategic business process because it automates and integrates other strategic business processes. It allows our customers to capture knowledge that is unique to their company and use it to sustain competitive advantage.

Over and over, IBM has seen customers use software development as a differentiator, and as a strategic advantage against the competition. By integrating content, processes, and tools from all of the IBM brands, and the services group, Rational is provided access to an incredible wealth of experience, capability, and tooling. IBM brings all of these to bear in the IBM Software Development Platform.

Q If every aspect of what developers want is now covered, from modeling through testing, what will be the future product cycle? How will the platform go on being improved still further?

Devlin: The focus is on deeper integration; seamless, bidirectional flow of information; and extending the cycle into the business domain and the operations domain. Rational is focused on knocking down walls between the traditional groups in IT.

For example, how long does it take to fix a bug that is discovered in production? A week? A month? Perhaps longer? By focusing on automating the processes behind the point tools, customers can start to solve the real problems that have prevented them from becoming more responsive and flexible – problems of process, communication, and information accuracy. With IBM, we have the ability to approach this problem from all angles, and by moving our tools onto Eclipse, we now have a common, open framework at the point where these efforts meet.

Q What will be the impact of UML 2.0 when the specifications are finally complete?

Devlin: UML 2.0 will provide an industry-standard modeling language that is specifically designed to support model-driven development. When UML was first proposed in the mid-90s, the primary focus was to raise the level of abstraction so that both problems and solutions could be expressed using concepts that were much closer to the problem domain than the technology domain.

UML 2.0 has gone even further in this regard with enhanced capabilities to model complex system architectures, Web services, and business processes. However, with its more precise semantics, UML 2.0 also adds a much greater potential for increased levels of automation; things such as executable models, extensive automatic code generation, and formal verification and validation. After all, automation has traditionally been the most effective technological means, by far, for dramatically improving productivity and reliability.

Q What is IBM's take on the dichotomy between high-end enterprise developers and what one might call the "business developer" – what does IBM offer those who are not members of the technology elite and for whom model-driven development isn't a must-have, but who can create GUIs, write HTML and JavaScript, build workflows, execute services, make maintenance changes, etc.?

Devlin: It's IBM's view that there is a spectrum of developers, and that no one approach can meet the needs of every developer. We provide solutions for code centric, visual, model-driven, technical, and corporate developers. These tools are migrating to a common meta-model that will allow them to interoperate and seamlessly share information. This will allow developers on the same team to use a mix of development skills and styles yet all work in the same project context with the same lifecycle tools.

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

Use Crystal Reports 10 with your J2EE applications

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

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

Visual Designer simplifies data connectivity

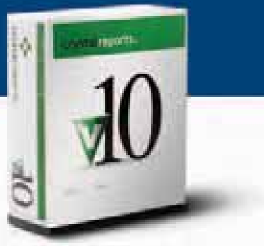
Deliver diverse data formatting options within your presentation layer

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

Expedite .NET and J2EE report integration

Design and integrate reports from within popular IDEs

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



New Crystal Reports 10.

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

Business Objects®

Building a J2ME Multimedia Messaging Service Client

by Shane Isbell

Host your own MMS content server and more

The Wireless Messaging API (WMA) reference implementation supports short message service (SMS) text and binary messages, but leaves the implementation of the hot area of multimedia messaging untouched. This article will demonstrate how to build a Multimedia Messaging Service (MMS) client using J2ME so you can get started writing new applications using this technology. By doing your own J2ME client implementation, you can bypass carrier lock-in of the content server or Multimedia Messaging Service Center (MMSC), as well as build your own server-side MMS applications, which you couldn't do before outside of the carrier network.

Software Requirements

A number of software requirements are needed to test the MMS client application. To run the basic emulation environment, make sure you have Sun's Wireless Toolkit 2.1 and JDK 1.4.2 or higher, both downloadable at http://java.sun.com. Download and compile the latest source code for the MMS Client and tools, available at http://sourceforge.net/projects/jvending. Next, set up either a Web or application server to act as the content server for the multimedia messages. Finally, you'll need to download the Nokia Mobile Internet Toolkit 4.0 from www.forum.nokia.com.

MMS Notification and Retrieval

The push registry maintains a list of inbound connections to mobile devices. It's one of the most exciting and least used features of the MIDP 2.0 spec. Under MIDP 1.0, a network-based application needed to constantly establish a connection and poll an application server for events. This ate up airtime and cost the user money, making network-based J2ME applications infeasible in many cases.

Shane Isbell works as a software architect at a wireless carrier.

random7@att.net

You can register inbound connections in the push registry by adding the connection information to the JAD file or by explicitly invoking the registerConnection static method on the PushRegistry class. In the J2SE world, registration is equivalent to opening a ServerSocket connection and waiting for a connection. The snippet below shows how we can register the connections in the JAD.

```
MIDlet-Push-1: serversocket://:1236,
org.jvending.messaging.mmclient.MmsMidlet, *
MIDlet-Push-2: datagram://:1235, org.jvend-
ing.messaging.mmclient.MmsMidlet, *
MIDlet-Push-3: sms://:1234,
org.jvending.messaging.mmclient.MmsMidlet, *
```

Why would you need to register so many push connections? There are a number of different ways to push the message notification, depending on the carrier network. You may need to set up push interfaces to cover push-over TCP (serversocket), WDP/UDP (datagram), and SMS. For instance, if we have an active GPRS or CSD connection, the server application may decide to send the MMS notification on the datagram connection, without failing over to the SMS connection. This means our MMS message is effectively lost.

Let's show how notification works in a carrier environment. Prior to the recipient MMS user agent retrieving an MMS message, the network must notify the user agent that the message is awaiting retrieval. There are a number of different ways for this notification to take place. The most common are SMS or WDP/UDP. If the user does not have a data session established, the notification will come over an SMS bearer. If a session has been established, the server just needs to push the request to the device over WDP/UDP (WAP 1.2+) or TCP (WAP 2.0).

Figure 1 shows a typical configuration within a carrier network for connectionless (asynchronous) notification through SMS. In step 1, the MMSC contacts the Push Proxy Gateway (PPG), which handles the pushing of messages to the mobile device. Next, the PPG contacts the Short Message Service Center (SMSC). In steps 3 and 4, the SMS is delivered over the GPRS network to the mobile device. If the MIDlet is not active, the application management software (AMS) detects that an SMS is destined, say, for port 1234. The AMS starts an instance of MmsMidlet, which extracts the MMS notification message from the body of the SMS message to find the MMS content location. In step 5, our application on the mobile device initiates an HTTP/GET connection to the content server.

Keep in mind that in most networks the connection is over the Wireless Session Protocol (WSP), which has an encoding that the content server does not understand. For steps 6 and 7, the

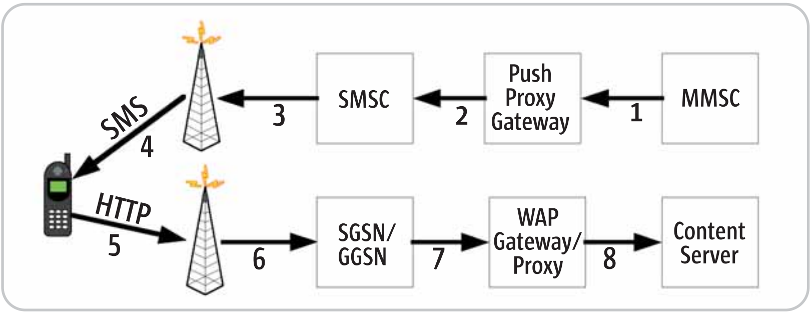


Figure 1 MMS retrieval

www.altova.com

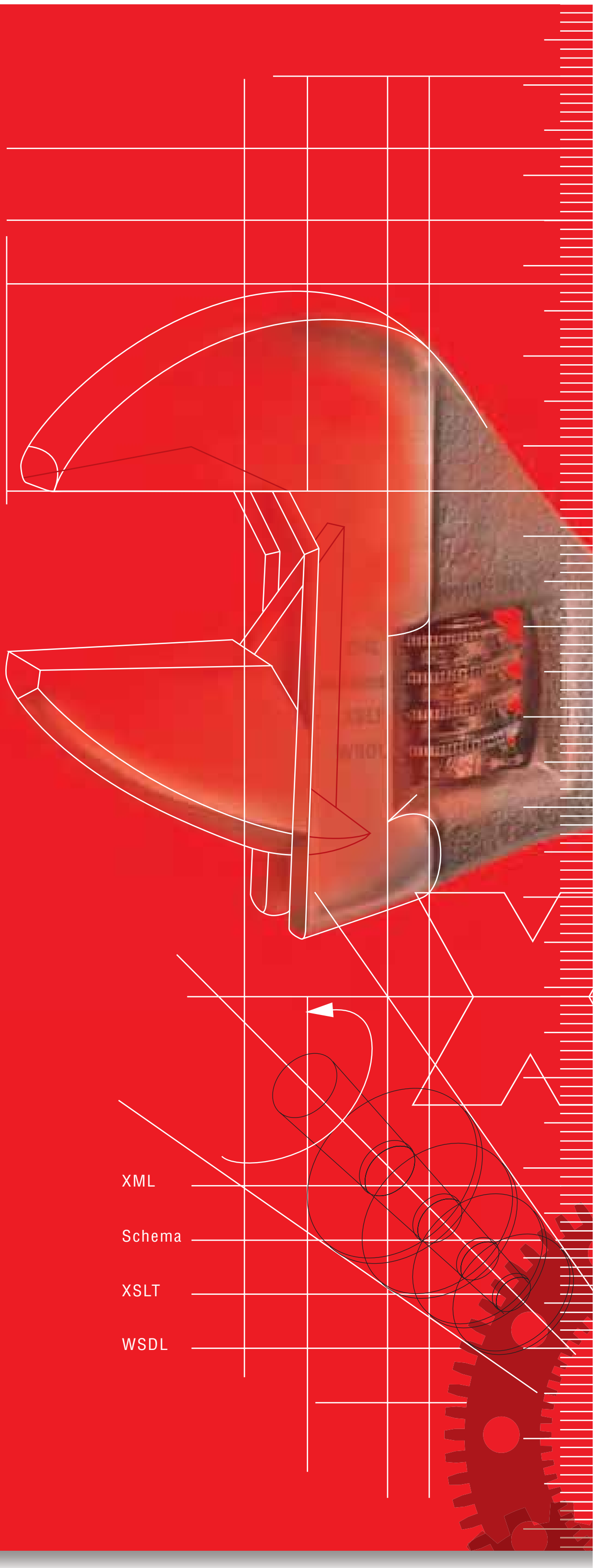
xmlspy® 2004

See what you can build:

Versatile design and editing views

Automatic code generation

Effortless XML differencing



xmlspy 2004 mapforce™ 2004 authentic® 2004 stylevision® 2004

*Look into **xmlspy**® 2004, the industry leading XML development tool from Altova. Discover why **xmlspy** 2004 is the standard for modeling, editing, transforming and debugging all XML technologies. With **xmlspy** 2004 developers can easily define XML Schemas and construct XML documents, then automatically generate runtime code in Java and other programming languages thereby saving time and accelerating development projects. See for yourself how **xmlspy** 2004 will build developer productivity.* ***Download xmlspy® 2004 today: www.altova.com***

ALTOVA®

www.altova.com

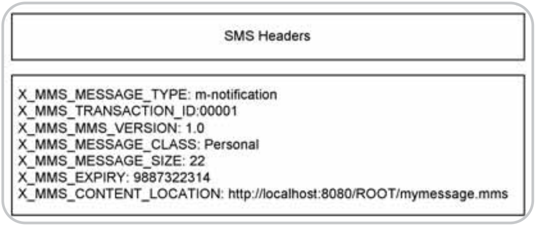
network routes the HTTP request to the WAP gateway. The WAP gateway translates the WSP binary encoded HTTP headers into normal HTTP headers and forwards the request (step 8) onto the content server. The content server now delivers the WSP encapsulated MMS message over HTTP back to the mobile device. The MmsMidlet receives the MMS message, decodes it, and displays the graphical content to the user.

Testing Message Notification and Retrieval over an SMS Bearer

Let's test out retrieving an MMS message. First we need to set up a content server, which is merely an application or Web server that stores the MMS content. You can find sample MMS files in the Nokia Mobile Toolkit. Place marketupdate.nosmil.mms on the application server at, say, http://localhost:8080/ROOT/marketupdate.nos mil.mms.

Now we need to generate an MMS message notification package. Do this by instantiating the org.jvending.tools .mms.MNotificationGenerator class, with parameters filename and content URL location (http://localhost:8080/ROOT/marketupdate.nosmil.mms), which points to the location of the MMS message on the content server. You now have an MMS notification message stored on your local file system.

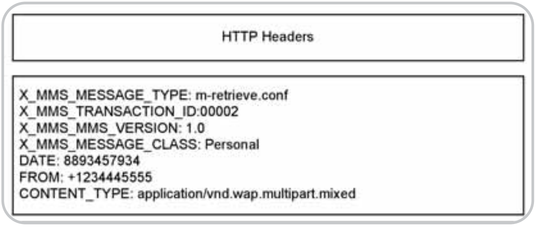
Next, open the WMA console within the Wireless Toolkit utilities tool. Make sure you're using version 2.1, since version 2.0 has a problem binary encoding characters within the 128–159 range. Click the "Send SMS" button followed by the Binary SMS tab. Import the contents of the MMS message that you generated with the MNotificationGenerator and type in port number 1234 within the text field.



SMS Headers

X_MMS_MESSAGE_TYPE: m-notification
X_MMS_TRANSACTION_ID:00001
X_MMS_MMS_VERSION: 1.0
X_MMS_MESSAGE_CLASS: Personal
X_MMS_MESSAGE_SIZE: 22
X_MMS_EXPIRY: 9887322314
X_MMS_CONTENT_LOCATION: http://localhost:8080/ROOT/mymessage.mms

Figure 2 MMS notification



HTTP Headers

X_MMS_MESSAGE_TYPE: m-retrieve.conf
X_MMS_TRANSACTION_ID:00002
X_MMS_MMS_VERSION: 1.0
X_MMS_MESSAGE_CLASS: Personal
DATE: 8893457934
FROM: +1234445555
CONTENT_TYPE: application/vnd.wap.multipart.mixed

Figure 3 MMS retrieve confirmation

Open up the emulator and click the MmsMidlet. You'll get a message asking if it's okay to receive text (or SMS) messages. Click OK. Go back to the WMA console and click "Send". This sends the binary SMS that contains the MMS notification to the recipient MMS user agent. Look at the package structure in Figure 2; the SMS body contains the MMS headers.

The MMS client application decodes this message and determines that it is an MMS notification message. The client reads the content URL location, opens an HTTP connection to the content server, pulls down the MMS message containing the mymessage.mms (see Figure 3), and displays the content on the mobile device. I'll explain exactly how to do this in the next sections.

Decoding the Multimedia Message

The MultimediaParser instance has a parse method that takes a PeekInputStream instance and a multimedia message object as parameters. A multimedia message object consists of two primary parts: the MMS headers and the MIME body (see Figure 3). The MultimediaParser instance delegates construction of the multimedia message object to two other objects, the HeaderFieldParser and the MimeParser.

The HeaderFieldParser's primary responsibility is to hand off the PeekInputStream object to an instance of WspTokenizer, which tokenizes the input stream into header fields according to the WSP spec. WSP encoding is a compact binary form that reduces the size of the headers.

The way this works is that if the first octet (or byte) is in the range of 32–127, the header is a text string, which ends with a 0 or null octet. If the first octet is in the range of 128–255, the header is binary encoded. For instance, if the value is 151, this maps to a TO field, which is subsequently followed by an encoded string. The less common values that begin in the range 0–31 are of variable length and center largely around date values.

This type of encoding makes tokenizing of the multimedia messaging surprisingly simple. You can decompose MMS into one of the following tokens: Text-String, Quoted-String, Extension-media, Short-integer, Multi-octet-integer, and Unitvar-Integer. The last two are integers of variable length, not something we have to deal with very often within the Java world. Essentially, they are just 128 base numbers and can be handled accordingly. See the source code to see how to encode/decode multi-octets. (The source code can be downloaded from www.sys-con.com/java/sourcec.cfm.)

With the exception of octets in the range of 0–31, the WspTokenizer object can determine the exact token and how to read the header based on the first octet. If the value is less than 32, however, the tokenizer peeks ahead one octet to determine the path that it needs to tokenize the header (see Listing 1).

After tokenization, the HeaderFieldParser object passes those tokens to an instance of the HeaderFieldAssembler, which takes the tokens and assembles them into easily readable Field objects to store within the target MultimediaMessage object.

Now that the HeaderFieldParser object has finished decoding the MMS headers, the MultimediaParser passes control to an instance of MimeParser, provided that the message contains a Content-Type field. The MimeParser object now assembles the MultipartEntry objects for the various MIME types and places them within the instance of MultimediaMessage. Our target MultimediaMessage object is now complete and ready to pass to the MessageConnection object, covered in the next section.

Extending the Wireless Message API

The wireless message API consists of five interfaces:

1. BinaryMessage
2. Message
3. MessageConnection
4. MessageListener
5. TextMessage

The BinaryMessage and TextMessage classes extend the Message interface. For this article, we add an additional class, MultimediaMessage, which also extends the message interface.

The javax.wireless.messaging. MessageConnection class has its own implementation on the mobile devices. Modifying this implementation to return MultimediaMessage objects would require recompiling core MIDP 2.0 classes. Thus, the MMS client has its own implementation called org.jvending .messaging.MessageConnection, which functions in the same way with the same API. Look at the receive method of org.jvending.messaging.MessageConnection. This implementation accepts two kinds of connections: HTTP and SMS, both of which return a Message object of type MultimediaMessage.

Jtest®

Verifying Algorithms Through Unit Testing

Complicated algorithms are typically contained within one class or function so that developers can easily manage them. How then, can you test the functionality of your algorithms and verify that they correctly solve problems before you have a full application written? The answer lies in unit testing.

Unit testing is the perfect strategy for verifying and testing algorithms. Unit testing involves testing software code at its smallest functional point, which is typically a single class. Each individual class should be tested in isolation before it is tested with other units or as part of a module or application. By testing every unit individually, most of the errors that might be introduced into the code over the course of a project can be detected or prevented entirely.

I suggest that you investigate the benefits of performing unit testing to verify the functionality of your algorithms. Performing thorough unit testing reduces the amount of work you will need do at the application level, and drastically reduces the potential for errors.

By the way, unit testing is a fundamental part of Parasoft's Automated Error Prevention (AEP) Methodology. You can find more information about Automated Error Prevention at **www.parasoft.com**

*– **Adam Kolawa**, Ph.D.*
Chairman/CEO of Parasoft

Automate unit test case generation for JUnit and Java with Parasoft Jtest®

Introducing Parasoft® Jtest®

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

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

Learn how Jtest can enhance JUnit capabilities...

Download a free eval copy of Jtest along with our informative new white paper entitled "Automated Java Unit Testing, Coding Standard Compliance, and Team-Wide Error Prevention."

For Downloads go to **www.parasoft.com/jtest**

Email: jtest@parasoft.com Or call 888-305-0041 x3303

PARASOFT® We make software work.™

FEATURES

- Fully integrated with JUnit and Eclipse
- Finds and fixes errors fast
- Automatically generates JUnit test cases
- Customizable testing and reporting

BENEFITS

- Makes error prevention feasible and painless, which brings tremendous quality improvements, cost savings, and productivity increases
- Makes unit testing and coding standard compliance feasible and painless
- Encourages the team to collaborate on error prevention

PLATFORMS

Windows 2000/XP
Linux
Solaris

Contact Info:

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

If the URL connection is HTTP based, invoking the receive method on the MessageConnection instance results in the mobile device initiating an HTTP connection to the content server and pulling down an MMS message. The MessageConnection object casts the connection as an HttpConnection and invokes the openInputStream method to obtain an InputStream object. This does not involve the WMA messaging functionality

If the connection is SMS, we leverage the WMA by casting the connection as javax.wireless.messaging.MessageConnection and receiving a BinaryMessage. We now need to get it into an InputStream object since this type is required to pass into the instance of MultimediaParser. This requires invoking the getPayloadData() method on the BinaryMessage and feeding this in as a parameter to an instance of ByteArrayInputStream.

Finally, the MessageConnection object creates an instance of PeekInputStream and then invokes the parse method on the MultimediaParser object. The target MultimediaMessage object is now filled with all the MMS headers and MIME entries. Note that the MessageConnection is not concerned with the type of MMS message or how to handle notification and retrieval flow. The MmsMidlet client handles this logic (see Listing 2).

The MMS Client

The org.jvending.messaging.mmsclient package contains two classes: the MmsMidlet, which contains the logic for instantiating MessageConnection classes; and the MultimediaViewer class, which handles the display of the headers and media types. The first thing the MmsMidlet does is open an SMS connection and then wait to receive a MultimediaMessage on port 1234 (of course, in the actual source code this is threaded).

When the MMS notification comes over the SMS bearer, the MmsMidlet strips out the body and confirms that it is an M_NOTIFICATION message type. The MmsMidlet object takes the content URL location from the MMS notification and uses it as a parameter when invoking the open method on the Connector class. As discussed in the previous section, invoking the receive method on the MessageConnection returns a MultimediaMessage object containing the MIME entries. Next the MmsMidlet passes the MultimediaObject to an instance of MultimediaViewer that will, in turn, display the media content on the user's device (see Listing 3).

Conclusion

In the wireless data area, innovation and development are often difficult because carriers tightly couple device applications and server-side services. When you use J2ME and J2EE together to build client and server applications, it enables you to bypass many of these constraints. All it takes is access to basic SMS and HTTP protocols.

This article illustrates this flexibility by showing how building a J2ME MMS client allows developers to host their own MMS content server. This article also covers the basics of MMS retrieval and notification. The source code contains an MMS encoder that allows the user to send MMS messages. When combined with the Mobile Media API (JSR 135), this provides the ability to create a lot of interesting applications. You can find the latest updates and complete source code at http://sourceforge.net/projects/jvending.

Resources

- Le Bodic, G. (2003). *Mobile Messaging Technologies and Services*. Wiley.
- Metsker, S.J. (2001). *Building Parsers with Java*. Addison-Wesley.
- *now.sms:* www.nowsms.com/messages/index.htm
- Open Mobile Alliance, OMA-MMS-ENC-v1_1-20021030-C: Multimedia Messaging Service Encapsulation Protocol version 1.1, October 2002.
- WAP Forum, WAP-230-WSP: Wireless Application Protocol, Wireless Session Protocol Specification Version 5, July 2001

Listing 1

```
int i = dis.readUnsignedByte();

if(i >= 0 && i <= 31) {
  tokenObject = new Integer(i);
  tokenType = Token.SHORT_LENGTH;
  tokenStateQueue =
    dataTokenizer.getTokenStateQueue(tokenState,
  dis.peekByte(1));
} else if(i == 31) {
   tokenObject = readUintvarInteger(dis);
   tokenType = Token.UINTVAR_INTEGER;
} else if(i == 34) {
   tokenObject = readString(dis);
   tokenType = Token.QUOTED_STRING;
} else if(i == 127) {
   tokenObject = readString(dis);
   tokenType = Token.TEXT_STRING;
} else if(i >= 32 && i <= 127) {
   tokenObject = readExtendedMedia(dis, i);
   tokenType = Token.EXTENSION_MEDIA;
} else if(i >= 128 && i <= 255)
   tokenObject = new Integer((i & 0x7f));
   tokenType = Token.SHORT_INTEGER;
   tokenState = i;
} …
Token t = new Token(tokenType, (Object) tokenObject);
```

Listing 2

```
if(urlConnection.startsWith("http:")) {
  httpConnection = (HttpConnection)
Connector.open(urlConnection);
  is = httpConnection.openInputStream();
} else if(urlConnection.startsWith("sms:")) {
  BinaryMessage bm = null;
  conn = (javax.wireless.messaging.MessageConnection)
  Connector.open("sms://:1234");
  bm = (BinaryMessage) conn.receive();
  byte[] data = bm.getPayloadData();
  is = (InputStream) new ByteArrayInputStream(data);
} else {
  throw new IOException("Unrecognized Message Type");
}
  PeekInputStream pis = new PeekInputStream(is);
  MultimediaMessage mm = new MultimediaMessage();
  MultimediaParser mp = new MultimediaParser();
  mp.parse(pis, mm);
```

Listing 3

```
MessageConnection conn=(MessageConnection)
  Connector.open("sms://:1234");
mediaMessage = (MultimediaMessage) conn.receive();
String header = mediaMessage.getHeader("X_MMS_MESSAGE_TYPE")
int messageType= MessageType.lookupMessageType(header);
String contentUrl =
mediaMessage.getHeader("X_MMS_CONTENT_LOCATION");

if(messageType == MessageType.M_NOTIFICATION) {
   conn = (MessageConnection) Connector.open(contentUrl);
   mediaMessage = (MultimediaMessage) conn.receive();
}

MultimediaViewer viewer = new MultimediaViewer(this,
mediaMessage);
viewer.displayView();
```

Windows

Java

OS X

AIX

Solaris

HP-UX

OS/400

MSI

Mobile

Script

Linux

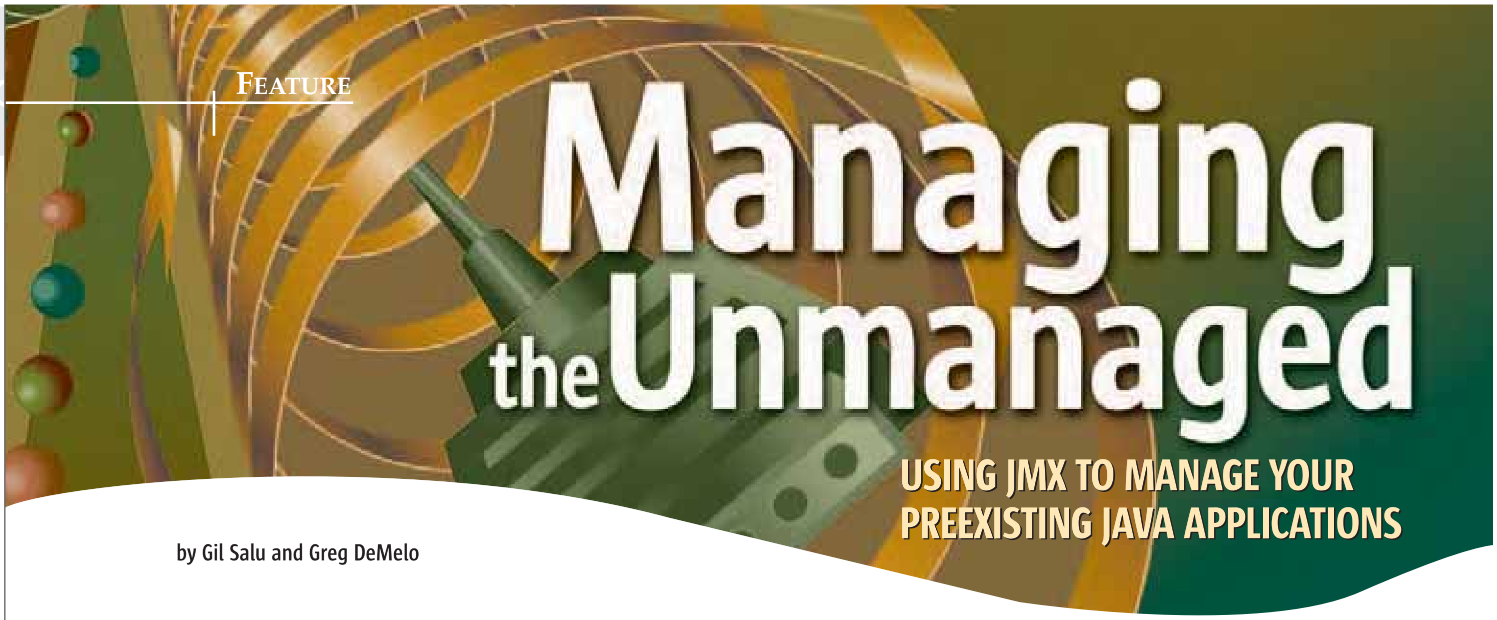
Redefining the Standard. *Everywhere*

InstallShield X

Download Now! www.installshield.com/redefine

From **Windows to Linux**, **Mobile to Servers**, your **One Solution** for Installations.

InstallShield
www.installshield.com



Managing the Unmanaged

Using JMX to Manage Your Preexisting Java Applications

by Gil Salu and Greg DeMelo

Hardly anyone ever thinks about application management frameworks until the application is running in a production environment and the application needs managing. The use of Java and J2EE has allowed business code to be written at lightning fast speed and application management is normally considered only as an afterthought. Without the convenience of an application server to start/stop and change various settings, these applications become monsters to administer and manage. Although JMX is a technology that is most often thought of for J2EE integration, JMX can also be used to quickly provide a management console for any well-designed standalone Java application with only minor refactoring efforts.

Gil Salu is an IT specialist with IBM Global Services' Enterprise Application Integration practice (www.ibm.com/services), where he develops system integration solutions. Gil is also a contributor to the ButterflyXML open source project (www.butterflyxml.org).

gsalu@us.ibm.com

Greg DeMelo is an IT architect with IBM Global Services' Boston Center for e-business Innovation (www.ibm.com/services/innovation/boston/). Greg has spent the last 8 years in the services industry focusing on custom application design and systems integration.

demelo@us.ibm.com

Sample Situation

A widget manufacturing company has a custom-built application that controls the rate of widget production. Every 10 seconds a new widget is produced. Most of the time the manufacturing line gets ahead of schedule and the workers sit idly by. Sometimes the line gets behind schedule at which point any onlooker would be reminded of Lucy and Ethel at the candy factory. When workers want to go to lunch, the application that controls everything must be killed. When lunch is over, the application must be manually restarted, a very lengthy process. Overall the production of widgets has become very inefficient.

Seeing these inefficiencies the business managers decide that they would like a way to change the rate at which widgets are produced. When the line is ahead of schedule, they'd like to fill capacity. When the line is slow and widgets are backing up, they would like to slow things down. Instead of killing the application, a system manager would be able to pause the application until everyone gets back from lunch. All of these functions should be administered from a nice management console that won't require too much training.

Charged with updating the system, the IT shop understands that an easy-to-adapt nonintrusive application management framework is needed to make their custom, standalone Java application manageable. The team decides that the JMX framework would be a perfect fit.

What Is JMX?

The specification for Java Management Extensions (JMX) addresses the need for a common framework for managing applications. It provides Java developers a much-needed management architecture, a set of APIs, and several management services that can easily be added to enhance any Java application. Currently JMX is used heavily in the J2EE application server world. As more and more software products become J2EE enabled, JMX is becoming the de facto standard framework for Java-based application management. It's not only J2EE services that can take advantage of JMX but also standalone Java applications that can also benefit. Developing these features may take place either during the initial build of the application, or, as the example presents, as an afterthought.

First let's get an understanding of the JMX architecture. The JMX architecture is three tier and easily integrated into any existing Java application. Figure 1 is adapted from the Sun Specification of JMX. It shows the three tiers and their components. It also shows where your preexisting application resides with respect to the rest of the management framework.

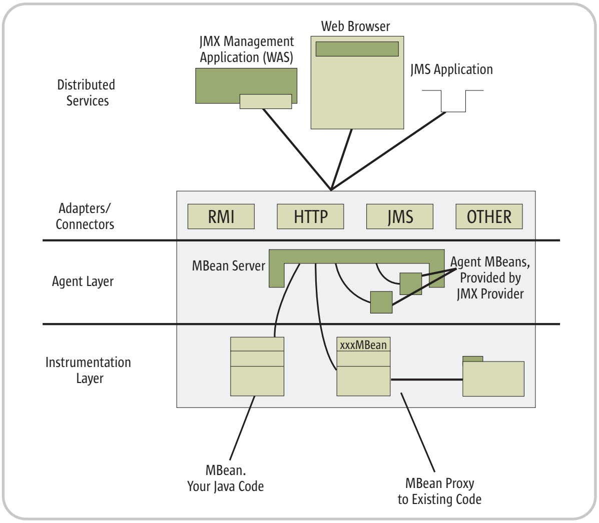


Figure 1 Architectural overview of the JMX framework (adapted from the Sun JMX specification)

The Instrumentation Layer

In the Instrumentation Layer, the Java developer exposes critical interfaces, objects, and components to the management interface by creating managed beans. JMX uses reflection to understand that an object is manageable, as such a managed bean (or MBean) is a class or interface whose name ends in the text "MBean", or a class that has in its hierarchy a class name ending in MBean (see www.sys-con.com/java/sourcec.cfm for more details). For example, a class Service would be considered an MBean if it was either renamed ServiceMBean or implemented an interface ServiceMBean.

The standard MBean allows each object's members to be exposed based on a certain set of rules.

- All attributes that have get methods are exposed for viewing in the management console.
- All attributes that have set methods are exposed for altering in the management console.
- All other public methods defined in the MBean are exposed as operations.

By creating and registering MBeans the instrumentation layer defines a core set of managed resources to the agent layer.

The Agent Layer

The JMX agent layer provides the magic of using the JMX framework. At the heart of the agent layer resides the MBeanServer. The MBeanServer acts as a repository and registry for all of a JVM's MBeans. The MBeanServer also provides many of JMX's services including event monitoring and timer services. The JMX agent provides the means to hook custom Java code into the JMX framework using the MBeanServer's register() method. The agent layer can be viewed as a black box to the Java developer since your JMX provider already implements the MBeanServer for you. JMX providers include JBoss, WebSphere, and, for this example, the Sun Reference Implementation of JMX. The agent layer uses a set of adapters and connectors that may be custom written or provided by your JMX implementation. These adapters and connectors act as a bridge between the agent layer and the distributed services layer.

The Distributed Services Layer

The last layer, Distributed Services Layer, provides the interfaces and components that remote tools use to interface with agents. Using the hooks provided by this layer, your JMX instrumented application can be accessed through HTTP, RMI, JMS, SNMP, or any other distributed protocol that makes sense to your application. Each JMX provider may supply adapters that make using these hooks even easier. In our example, the Sun Reference Implementation provides a simple, yet elegant, HTTP adapter that displays agent information through a set of HTML screens. (The source code can be downloaded from www.sys-con.com/java/sourcec.cfm.)

What to Do?

Now that we know what JMX is and how it can help us, let's walk through the steps involved in making your existing standalone Java application manageable through a simple HTTP service.

Step 1: *Figure out what needs to be managed. What does management mean to your application?*

Figuring out what needs to be done will lay a solid groundwork for understanding the best approach to exposing application functionality. This process will determine what components reside in your instrumentation layer. Our example involves a server that runs when a message arrives in the system and waits with a time interval for another message to come. We want to add the ability to manage the wait time. We would also like to be able to pause and resume this process as well as start and stop the thread from an object perspective. In our example we have a preexisting class named Service. Service has the main method for our application. It spawns a thread that checks the status of the object (started, stopped, etc.). In a loop, the server thread then calls the process() method.

Step 2: *Find the application code that will help expose management functions.*

Many of the management requirements will be able to be satisfied by simply exposing existing code. New code would be used to expose logic that wasn't exposed before. In our example we would like the Service class to be exposed to a management console. Since the existing Service class has all the methods of interest already implemented, we don't have to implement new application code (see Figure 2).

"The specification for JMX addresses the need for a common framework for managing applications"

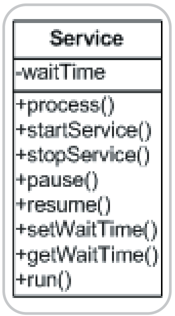


Figure 2 Preexisting Service class

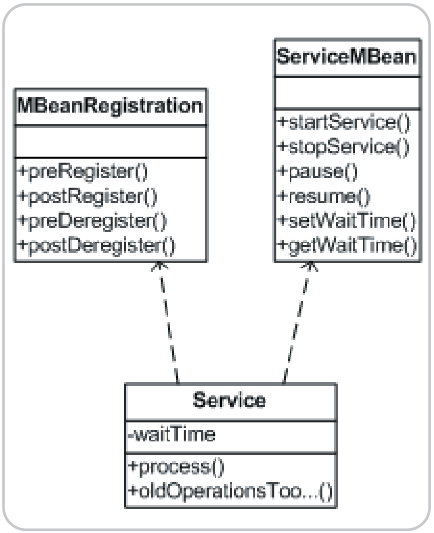


Figure 3 MBean instrumentation of the Service class

“Allow developers to quickly extend application components to deliver a simple yet functional management console ”

Step 3: *Write proxy MBeans that expose these functions.*

Once you know which methods you want to expose to the management console, develop MBean interfaces and make your classes implement them. MBeans are simple to write since they shouldn’t contain any of the logic that you want to expose. For our example we create a new interface: ServiceMBean. ServiceMBean defines the methods that we wish to expose. Note that “MBean” in the name is needed. The Service class must also now implement MBeanRegistration’s methods. These implemented methods may be empty since we don’t want to override any default behavior. Our original class Service should now implement ServiceMBean (see Figure 3).

Step 4: *Write the code that will register MBeans with the MBean Server.*

MBean objects need to be registered with an MBean Server. Registering them is a straightforward task. For our example, Service acts as the main class. The main method uses code in Listing 1 to create the javax.management.MBeanServer and register our instance of Service to it.

Step 5: *Provide the MBean Server to a distributed adapter.*

For our example, we use the Sun Reference Implementation of JMX, which provides us with an HTTP adapter and server and is found in jmxtools.jar. Using this adapter is a perfect way to quickly expose your MBean interfaces. The implementation provides an HTML screen that lets you browse and edit your MBeans; it is perfect for quick implementations. Again in our Service class’s main method we put the code shown in Listing 2.

Step 6: *Manage your application.*

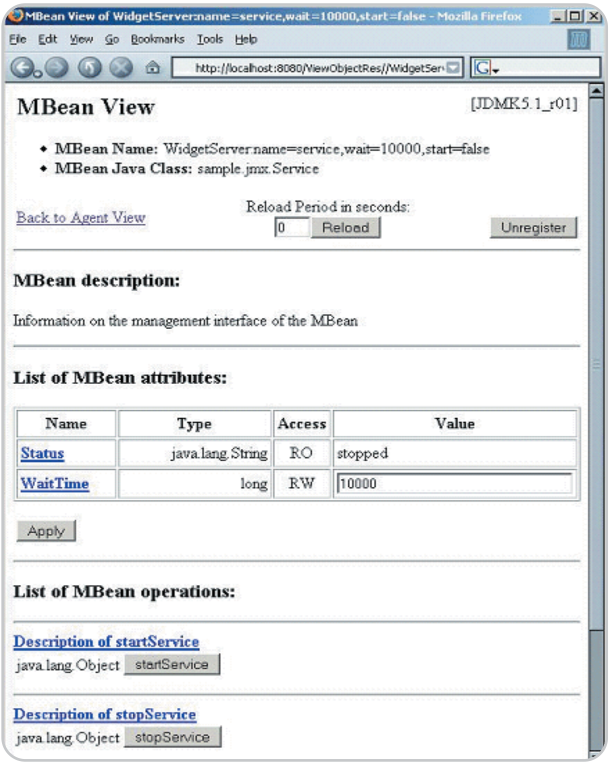
Now that you have written and registered all of your MBeans, and instantiated and started your adapter, you are ready to manage your application. In our example we can simply point a browser (Mozilla, IE, or even Lynx) to localhost:8080 and begin to manage our application. The Sun RI provides an intuitive user interface that allows each MBean’s attributes to be changed and its methods invoked. Figure 4 shows what comes with the Sun RI.

Conclusion

Using JMX as described can allow developers to quickly extend application components to deliver a simple yet functional management console. The example shown here shows the power of simply adding MBeans and JMX to your standalone application. The HTML adapter that is provided with the JMX Reference Implementation is enough to get a standalone Java application manageable in a short period of time. The JMX world becomes even more powerful when other adapters and distributed services are used. By using these and other features of JMX, more complex and robust solutions are easily contrived.

Resources

- Perry, J.S. (2002). *Java Management Extensions*. O’Reilly: www.oreilly.com/catalog/javamngext
- Sun JMX Specification and reference implementation: http://java.sun.com/products/JavaManagement


MBean View of WidgetServer:name=service,wait=10000,start=false - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/ViewObjectRes//WidgetSer

MBean View [JDMK5.1_r01]

- **MBean Name:** WidgetServer:name=service,wait=10000,start=false
- **MBean Java Class:** sample.jmx.Service

Back to Agent View Reload Period in seconds: 0 Reload Unregister

MBean description:

Information on the management interface of the MBean

List of MBean attributes:

Name	Type	Access	Value
Status	java.lang.String	RO	stopped
WaitTime	long	RW	10000

Apply

List of MBean operations:

Description of startService
java.lang.Object startService

Description of stopService
java.lang.Object stopService

Figure 4 The HTMLAdapter interface showing the newly instrumented Service class

Listing 1

```
//Create MBeanServer
MBeanServer server = MBeanServerFactory.createMBeanServer();
System.out.println("\n\Creating and Registering Service");
ObjectName serviceName = null;
try {
   Service aService = new Service();
   serviceName = new ObjectName("Domain:name=service");
   server.registerMBean(aService, serviceName);
} catch (Exception e) {
   //Couldn't register
   e.printStackTrace();
}
```

Listing 2

```
//create the HTMLAdapter
HtmlAdaptorServer html = new HtmlAdaptorServer(8080);
ObjectName htmlAdapterServerName = null;
try {
   htmlAdapterServerName= new
      ObjectName("Adaptor:name=html,port=8080");
   //Since the HTMLAdaptor is itself JMX compliant, we
   //register it. This gives it access to registry
   //information that exists inside the  MBeanServer
   server.registerMBean(html, htmlAdapterServerName);
   //start listenting
   html.start();
} catch(Exception e) {
   System.out.println("\nError Creating the html adapter");
                    e.printStackTrace();
}
```

< Know More. No Less. >

Knowledge.
Experience.
Confidence.
From Our Team to Yours

TRAINING
Innovative,
custom-tailored Courses
to maximize your ROI

- Object Oriented Analysis & Design
- Requirements, Modeling and MDA™
- Java™, J2EE
- WebSphere™ & WebLogic™
- .NET™
- XML and Web Services

MENTORING
Building confidence,
delivering results

- Senior, real-world architects & developers
- A+ educators with advanced degrees
- Fortune 1000 customers
- 30,000+ technical professionals trained

CONSULTING
Technical mastery.
Commitment to excellence.
At work for you.

- Enterprise Architecture
- Application development
- Software Migration
- Application Integration
- Technology Assessment
- Team & Process Assessment

Know More.
www.inferdata.com for a 10% discount!

InferData

Building the Ultimate Logging Solution

...and solving common interoperability issues

by Jerason Banes

Anyone who has dealt with complex enterprise applications knows the value of a good logging solution. Features such as consolidating log files, separating events, and turning debugging on or off all come free with a good logging API. As a result, developers have been converting their existing System.out logging to many of the advanced solutions that have appeared on the Java scene.

This article attempts to demonstrate how to convert your System.out logging to the new Java 1.4 logging APIs. Along the way, new concepts and algorithms will be introduced to solve common interoperability issues.

Using the 1.4 Logger API

The key to Java 1.4 logging is the java.util.logging.Logger class. Instances of this class get created on demand and may be statically referenced for the life of the application.

While many applications will only deal with a single Logger instance, it may make sense for some applications to have multiple loggers. An example of this is a job scheduling server that creates a separate log file for each task (see Figure 1). To split log messages between files, the server assigns a new Logger instance when the task is created. The result is a much cleaner separation of events – something that can be invaluable when tracing a problem.

To obtain a Logger instance, we need a unique name. Keep in mind that loggers exist for the life of the application, so it's very important to choose a unique name. Failure to do so can result in strange or unexpected behavior. Since class names must be unique, a common solution is to use the primary class name. Thus a Logger for the class com.example.MyServer might be obtained via the following code:

```
Logger logger = Logger.getLogger("com.exam-
ple.MyServer");
```

Log Records Explained

Sending a message is a little different than printing to an OutputStream. Each logging message needs to contain the text of the message and the "error level" of the message. The "level" is used by the Logger to decide if the message should be discarded. This feature is often used to turn debugging information off in production environments.

To log a message, we need to create a new instance of the java.util.LogRecord class. This class encompasses all the necessary information about the message, including the text and the "level." Once created, the object can be passed to the Logger's log() method, which will either output the message or discard it.

The following example prints "This is an informational message." to the log:

```
LogRecord record = new LogRecord(Level.INFO,
"This is an informational message.");
logger.log(record);
```

Levels of Logging

As previously mentioned, each LogRecord has a "level." While you might be tempted to think that this equates to the "type" of the message, it doesn't. In fact, individual logging levels cannot be turned on or off. A Logger instance can instead be configured to only print messages above a certain level. All messages below that level are ignored.

Logging levels are defined by the java.util.logging.Level class. This class wraps an integer that defines the current level. While you can create new instances of Level class, it's generally much easier to use one of the predefined levels.

The predefined levels (from highest to lowest) are as follows:

```
Level.SEVERE
Level.WARNING
Level.INFO
Level.CONFIG
Level.FINE
Level.FINER
Level.FINEST
```

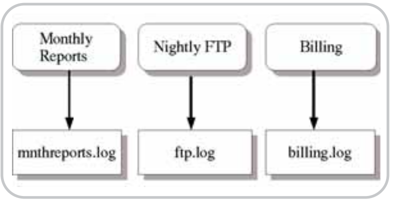


Figure 1 Job scheduler log

FINE, FINER, and FINEST are usually for various levels of debugging information and would normally be turned off in a production environment. CONFIG is for printing information on the current server configuration; INFO is for informational messages such as those usually emitted via System.out; WARNING is for possible problems; and SEVERE is for outright program errors. Custom levels are usually higher than SEVERE.

The default is to log Level.INFO or higher. This can be changed by making a call to Logger.setLevel(), or by modifying the "lib/logging.properties" file in the JRE directory. Look up java.util.logging. Log Manager in the Javadocs for more information on modifying the logging.properties file.

Here's an example of using the FINEST level to log debugging info:

```
Logger logger = Logger.getLogger("com.exam-
ple.MyServer");
LogRecord record = new
LogRecord(Level.FINEST, "Value of 'myvar' is
"+myvar);
logger.log(record);
```

Redirecting to a File

While the ability to control which text gets logged is a useful feature unto itself, no logging solution would be complete without a straightforward method for logging to a file. Java 1.4 logging is no exception.

After checking the logging level, the Logger class calls a subclass of java.util.logging.Handler. The Handler is responsible for writing the LogRecord to the console, a file, or even a network stream. For the purposes of this article, we'll be using the java.util.logging.FileHandler class to log to a file.

To use the FileHandler class, we need to create a new instance and call Logger.setHandler(). The name of the log file can be specified by passing a string to the FileHandler constructor. Here is an example of logging to a file named "test.log":

```
Logger logger = Logger.getLogger("com.exam-
ple.MyServer");
Handler handler = new
FileHandler("test.log");
LogRecord record = new
```

Jerason Banes is a long-time Java developer and architect who enjoys Java architectural challenges. He currently is spending his time working on a cross-platform, cross-vendor database interface product known as DataDino Database Explorer (). On the rare occasions that he's not programming, he spends his time with his wife and two children.

jbanes@techie.com

```
LogRecord(Level.FINEST, "Value of 'myvar' is
"+myvar);

logger.setHandler(handler);
logger.log(record);
```

Formatting for Nicer Output

If you tried the examples above, you probably noticed that the logging output isn't the cleanest. Sun built many of the default settings for the logger to be as situation independent as possible. While I'm sure that Sun had everyone's best interests in mind, most situations call for a more condensed format. The most common choice is a single-line format that allows for quick browsing and easy log analysis with common tools like "grep."

To demonstrate what I mean, here's the output from an earlier example:

```
Jan 26, 2004 12:04:34 AM JavaLoggerTest main
FINEST: Value of 'myvar' is ABC123
```

Using this format, we end up with two lines in the log for every one line we output! In theory, there might be circumstances where we write multiple lines in a single logging message (such as Stack Traces). In practice, most of our log records will be no more than one line.

To correct this, we'll configure the logger so that each line of the log message is prefixed with the message info. Since my preferred format is [yyyy-MM-dd HH:mm:ss.S] <LEVEL>: <Message>, our example will use that as the format.

Formatting is handled by subclasses of the java.util.logging.Formatter class. By overriding the format(LogRecord) method, we can develop a custom look to our logs. Formatters are attached to Handlers so that the format can be customized based on the destination. See Listing 1 for our example format.

After adding in our new formatting class, we now have the following example code:

```
Logger logger = Logger.getLogger("com.exam-
ple.MyServer");
Handler handler = new
FileHandler("test.log");
Formatter formatter = new
SingleLineFormatter();
LogRecord record = new
LogRecord(Level.FINEST, "Value of 'myvar' is
"+myvar);

handler.setFormatter(formatter);
logger.setHandler(handler);
logger.log(record);
```

The output has now changed to look like the following:

```
[2004-01-29 20:36:53.705] FINEST: Value of
'myvar' is ABC123
```

As you can see, the output is now much cleaner and easier to read. It also sets the stage for sending multi-line output one line at a time. This will be important later when we attempt to redirect OutputStreams to our log.

Converting Old Code

Our examples so far have assumed that we're only logging a single message, and as a result have performed various steps that would be redundant in a real application. I'm going to show a more "normal" approach by demonstrating how to convert code that uses System.out and System.err into code that uses the logger.

The secret to adding proper logging to your application is configuring the Logger ahead of time. Once the configuration is out of the way, System.out statements can be easily replaced with similar one-line code statements. Let's take the lessons from above and add the proper setup code to a main() method:

```
public static void main(String[] args)
{
    ...
Logger logger = Logger.getLogger("com.exam-
ple.MyServer");
    Handler handler = new
FileHandler("test.log");
    Formatter formatter = new
SingleLineFormatter();

    handler.setFormatter(formatter);
    logger.setHandler(handler);


    ...
}
```

An alternative is to modify the "lib/logging.properties" file. In fact, the logging.properties file is how most "real" applications commonly handle the setup of the Loggers. I won't cover the format of the file in this article, but it's well documented in the Javadocs for the LogManager class.

Now that we've configured our logger, we can convert some code. Here's a typical example of the type of code we'll be converting:

```
try
{
    System.out.println("This is a debug
statement.");
    System.err.println("This is an error
statement.");
}
catch(Exception e)
{
    e.printStackTrace();
}
```

The first output statement is pretty easy to convert based on what we know:

```
Logger.getLogger("com.example.MyServer").log(
new LogRecord(Level.FINEST, "This is a debug
statement."));
```

The second output statement is more of the same, with the primary difference of a higher logging level:

```
Logger.getLogger("com.example.MyServer").log(
new LogRecord(Level.SEVERE, "This is an error
statement."));
```

The third and final output statement is far more interesting. Many programmers would be tempted to log the message of the exception but continue to send the stack trace to the standard error stream. Thankfully there's a better way.

The Sun developers recognized the need to log exceptions and thus added a special Logger.log() method. This method takes the standard LogRecord parameter, but adds a new parameter of type Throwable. Since all exceptions and errors extend the Throwable class, all exceptions can be logged with this method.

The final converted code looks like Listing 2.

What Are Thread Locals?

The MultiOutputStream class uses a new instance of java.lang.ThreadLocal to associate each OutputStream with a thread. But how does it work? The answer is hash tables. Consider the following code:

```
public class ThreadLocalHashtable extends Hashtable
{
    public void set(Object value)
    {
        Thread thread = Thread.currentThread();

        super.put(thread, value);
    }

    public Object get()
    {
        Thread thread = Thread.currentThread();

        return super.get(thread);
    }
}
```

Notice how the "set" method is passed a value Object, but no key Object? That's because the the Thread.currentThread() method is used to find the current thread. That Thread object is then used as a key to the hash table. This allows a thread to store an object that cannot be changed or accessed by any other thread.

Tip: InheritableThreadLocal

ThreadLocal has a subclass called Inheritable-ThreadLocal. This subclass enables a child thread to inherit a local from its parent thread. When dealing with pluggable code like servlets, it's often desirable to assign the Logger to any child threads created by pluggable code. Without this precaution, a System-.out call may result in a NullPointerException.

Dealing with Libraries

Now that we've converted all of our code to using a logger, we're still left with the difficulty of code that isn't owned by our project, i.e., Libraries.

In a perfect world, there would be an easy way to make all libraries use our logger. There are a few things we can do, though. In the following sections I'll cover a way to interface System.out and System.err with our logger.

Redirecting the Standard Output Streams

To accomplish our goal, we'll need a custom OutputStream that creates a new LogRecord object every time a line break is received. We can then wrap the OutputStream in a PrintWriter object, and call System.setOut() and System.setErr() to replace the standard output streams.

While you may feel free to write your own solution, I think you'll find that the class in Listing 3 will meet your needs quite nicely. Despite its size, the class is actually rather robust and will work for replacing both System.out and Systemerr. This class only needs to be installed once at the beginning of your program.

The code below demonstrates its usage. You'll note that I've assigned System.out to the Level.INFO level and System.err to the Level.SEVERE level. This allows the two streams to be easily distinguishable in the log file.

```
public static void main(String[] args)
{
    ...

    System.setOut(new PrintWriter(new
LoggerOutputStream("com.example.MyServer")))
;
    System.setErr(new PrintWriter(new
LoggerOutputStream("com.example.MyServer",
Level.SEVERE)));

    ...
}
```

Assigning Multiple Loggers to the Standard Streams

Now that we've solved the problem of integrating the standard output streams with our logger, we've managed to create a new problem. For most applications, a single logger is sufficient. However, comprehensive servers (such as a Web application server) are usually configured with a separate logger for each application the server deploys. Our solution above would require you to choose a single log file for the standard streams, and stick with it.

One way of solving this is to use a ThreadLocal object. By associating a thread to its Logger, we can effectively create a lookup table of the output streams to be used.

Let's take our previous example of a job scheduling server. If we assume that each task is run inside its own thread, we can assign a LoggerOutputStream to the task when it's created. Obviously, this solution wouldn't work if we decided not to use multithreading, but it should be workable under most circumstances.

While not a perfect solution, it solves 90% of the cases where we use multiple loggers. See Listing 4 for code that demonstrates this concept. (Listings 4 and 5 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

Listing 5 shows how our fictional job scheduling server might use the Multi-OutputStream class to associate Loggers with tasks. It's important that the code to associate the OutputStream happens as soon as the Thread is created. If you wait, you may run the risk of a NullPointer-Exception.

Final Thoughts

While this article has (hopefully) served as a good introduction to logging, it is in no way comprehensive. Many extremely useful features have been either omitted or glossed over for the sake of simplicity. To obtain more information on these features, I would highly recommend that you read the Javadocs or pick up a good book on logging.

Listing 1

```
import java.text.*;
import java.util.*;
import java.util.logging.*;


public class  SingleLineFormatter extends Formatter
{
    private SimpleDateFormat dateformat = new
     SimpleDateFormat("yyyy-MM-dd HH:mm:ss.S");
    private Date date = new Date();

     public String format(LogRecord record)
     {
         date.setTime(System.currentTimeMillis());

         return "["+dateformat.format(date)+"]
"+record.getLevel()+": "+record.getMessage()+"\r\n";
     }
}
```

Listing 2

```
try
{
    Logger.getLogger("com.example.MyServer").log(new
LogRecord(Level.FINEST, "This is a debug statement."));
    Logger.getLogger("com.example.MyServer").log(new
LogRecord(Level.SEVERE, "This is an error statement."));
}
catch(Exception e)
{
    Logger.getLogger("com.example.MyServer").log(new
LogRecord(Level.SEVERE, "An error occurred!", e);
}
```

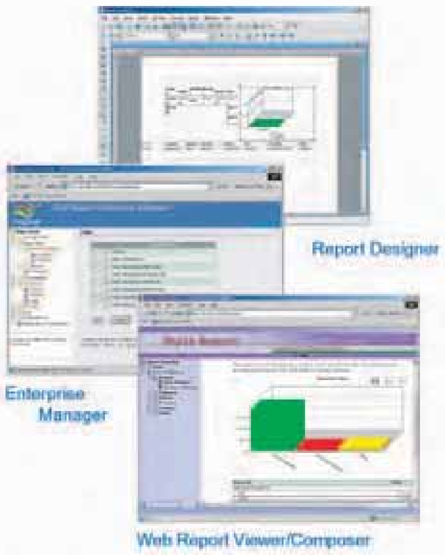
Listing 3

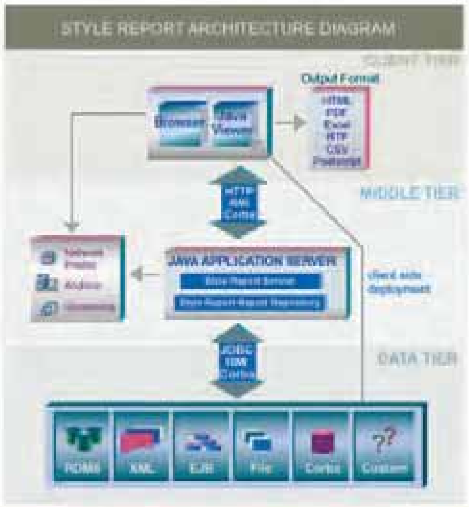
```
import java.io.*;
import java.util.logging.*;

public class LoggerOutputStream extends OutputStream
{
    private Logger logger;
    private StringBuffer buffer = new StringBuffer();
    private Level level;

     public LoggerOutputStream(Logger logger)
     {
         this(logger, Level.INFO);
     }

     public LoggerOutputStream(Logger logger, Level
       level)
     {
         this.logger = logger;
         this.level = level;
     }

     public void write(int c) throws IOException
     {
         if(c == '\n')
         {
             logger.log(new LogRecord(level,
                buffer.toString()));
             buffer = new StringBuffer();
         }
         else
         {
             buffer.append((char)c);
         }
     }
}
```

Java & Web Reporting:
with Java flexibility
without Java complexity

Style Report:
Information Vantage Point

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





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

JAVA DEVELOPER'S JOURNAL JDJ READERS' CHOICE AWARD

InetSoft
open standards innovation

www.inetsoft.com
JDJ special http://www.inetsoft.com/jdj

Phone: 888-216-2353 (USA)
Phone: 732-424-0400 (Intl)

PREVENTING REVERSE ENGINEERING

Java Bytecode Obfuscation

by J. Steven Perry

Your software may be under attack! Reverse engineers can't wait to get their hands on your binaries and learn their secrets. Okay, so that's a little melodramatic, but for many software companies like mine, there is a competitive advantage in our source code, so we don't ship source, rather only the necessary artifacts required to execute on the target platform. While an extremely motivated individual, given enough time, energy, patience, and Mountain Dew, can reverse engineer the software by disassembling the execution artifacts down to the machine-code level and figuring out how the software works, shipping only software executables is pretty safe.

What about software written in Java? Java source is compiled into .class files, which are often packaged into Java Archive files (JARs) and shipped, along with any Javadocs and other documentation. The situation is no different, really, from software that is compiled into native machine language instructions, right? Well, not exactly, as we will see.

In this article, I'll explore the vulnerabilities of Java bytecode to decompilation-style reverse-engineering attacks. Then we will look at a technique called obfuscation for modifying the bytecode instructions so that, if subject to such an attack, the resulting decompiled code is more difficult to read. We'll see how to run an obfuscator and explore some of the options available for obfuscating bytecode. Finally, we'll look at some of the things to keep in mind when using an obfuscator.

The source code for this article is available at www.sys-con.com/java/sourcec.cfm, and contains a complete working application that I wrote for chapter 10 of the book *Java Enterprise Best Practices*. Scripts to build and run the application, along with scripts to decompile the .class files, are also included.

J. Steven Perry has 13 years experience as a professional software developer, and for the past five of those Steve has been focused on Java development in such areas as application management, XML data binding, and enterprise frameworks. He is an author, a participant on two Java Community Process Expert Groups, and works as an architect for Fidelity Information Services in Little Rock, AR.

steve.perry@fnf.com

Introduction

Lately I've been thinking a lot about Java bytecode, the instructions produced by the Java compiler and executed by the Java Virtual Machine. Let's suppose I have a class, Queue, whose add() method is shown in Listing 1.

When the Java source code for this class is compiled, a file is produced with a .class extension that contains some metadata about the class, along with bytecode instructions for executing the class's instructions. I can use a decompiler such as JODE (http://jode.sourceforge.net) to decompile the Queue class. The add() method is shown after decompilation in Listing 2. In this listing, Queue.java was compiled with debug information included and no optimization.

Notice something rather startling: nearly all of the original Java code can be reproduced from the contents of the class file! With the exception of the comments from Listing 1, the original Java code and the code produced from the decompiler are identical. But since I never (oh no, not me!) forget to change my <javac> Ant task debug flag in my build script (or omit the –g flag when compiling from the command line), I've got nothing to worry about, right? Let's see. Listing 3 shows the decompiled add() method from Listing 2 when debugging information is not included in the class file. In Listing 3, Queue.java was compiled with no debug information.

Again, with the exception of comments and local variables in the add() method, the original source code has survived. Although the local variable names add meaning when reading the code for the method, it still wouldn't be terribly difficult to reverse engineer this code.

What do I do to protect my software from reverse engineering? Of course, I use an obfuscator! There are several freely available bytecode obfuscators. The working application that accompanies this article uses two obfuscators: ProGuard (http://proguard.sourceforge.net/) and yGuard (www.yworks.com/en/products_yguard_about.htm), both freely available (ironic? see sidebar – Reconciling Open Source with Obfuscation). Of the many available, I picked these two because they integrate with Ant, which is my build tool of choice. The examples in this article show the yGuard obfuscator in action.

Concepts of Obfuscation

The main idea behind bytecode obfuscation is to take a Java class file and process it into a new class file. By doing so, the new class file is behaviorally identical to the original, but bytecode instructions and class file metadata are scrambled so that reading and understanding decompiled obfuscated bytecode is difficult. Ideally, all obfuscating transformations on the original bytecode should be one way, or lossy. That is, the process so completely scrambles the bytecode that the bytecode still executes as intended, but

Simplifying Integration

Grid-Enabled Java Messaging

Service Oriented Architecture

Event-Driven Business Processes

Composite Application Platform

Standards-Based

Fiorano Middleware

Enabling the Event-Driven Enterprise

- Fiorano Business Integration Suite ™
- Fiorano Enterprise Service Bus ™
- FioranoMQ ™

Fiorano

Learn more at www.fiorano.com

unscrambling it with a decompiler retrieves very little of the original source. We've already seen how the bytecode produced by the Java compiler can be easily reproduced (especially if we compile with debug information included in the class file). An obfuscator can employ several techniques to foil the would-be reverse engineer. In the following section we'll look at the simplest of those techniques: layout obfuscation.

Layout Obfuscation

Layout obfuscation refers to altering the formatting of the class file. This involves removing debug information and changing the names of elements such as the class, member variables, and the local variable.

Remove Debug Information

Of course, debugging information can be omitted by the way you compile the code, but an obfuscator offers this initial level of protection should you forget. When code with debugging information in it is decompiled, local variable names are preserved. Any proprietary algorithms contained in the code can then be easily reverse engineered.

Renaming

The obfuscator employs renaming techniques to further confuse the would-be reverse engineer. Renaming is a powerful obfuscation technique. Why? Properly written, there is a good deal of semantic information contained in the names of classes, methods, and variables used in source code. Removing the inherent meaning in the class, member, and local variable names and replacing them with names that are not related to their purpose at execution time results in far less readable code, as shown in Listing 4. (Listings 4–6 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

As you can see, this code is pretty hard to read. The method name along with class member and local variable names have been replaced with short, meaningless names. This is the kind of thing an obfuscator does: it makes your Java bytecode less susceptible to reverse engineering.

In fact, I also configured the obfuscator to rename the Queue class, all member variables, and certain member methods [as we saw in Listing 4 that add() was renamed to A()].

In Listing 5 we can see that Queue was renamed to C, and its base class (Basic) was renamed to B. All of the member variables were also renamed. The structure of the Queue class is essentially the same, but all of the meaning I coded into the names of variables, methods, and the class name is gone. And, best of all, this is a one-way transformation, so the original semantics of the class are lost upon decompilation, as we see from the previous examples.

Things to Look for in an Obfuscator

Not all obfuscators are the same, but there are some commonalities between obfuscators, and for good reason. Any obfuscator should be able to remove debug information and rename identifiers. However, a well-written obfuscator should also be configurable so you can pick and choose which identifiers are preserved and which are obfuscated. A good obfuscator will also provide some sort of log file that contains the mappings from original names to obfuscated names (some obfuscators even have separate tools to make looking this information up easier) so that you can, for example, interpret stack traces.

Here is a laundry list of the minimum functionality an obfuscator should provide:

- Remove debug information
- Rename identifiers to be meaningless
- Configurable renaming so that you can choose what gets renamed and what gets obfuscated
- Generate a mapping file so you can map original names to obfuscated names

Control Obfuscation

Another powerful obfuscation technique is Control Obfuscation, which refers primarily to altering the control flow of the statements that execute inside a method. This is a very sophisticated technique, and one that I could only find implemented in commercial obfuscators. An obfuscator that implements this technique produces bytecode for a class whose method instructions are altered such that the method still executes as intended. However, should the resulting class be decompiled, the code is even more difficult to decipher than it is by using renaming techniques.

Control flow obfuscation should be used with care, however, because when the flow of a method is altered, the potential to introduce overhead becomes very real. While a top-of-the-line obfuscator will certainly take this into consideration, it would be wise on your part to benchmark your unobfuscated code against your obfuscated code, especially if your obfuscator aggressively alters control flow. Some commercial obfuscators make the level of control flow obfuscation configurable, from none to aggressive.

An in-depth discussion of control flow obfuscation is found in "A Taxonomy of Obfuscating Transformations" by C. Collberg, et al.

Running the Obfuscator

All of the examples in this section use the yGuard obfuscator, which also produced the examples we've seen so far. Every obfuscator I've worked with has a slightly different configuration, but basically configuration falls into general categories, which we'll look at below. However, the configuration shown is that of yGuard, so you can get started with the example application, which can be downloaded from www.sys-con.com/java/sourcec.cfm. All configuration is in XML, since we'll be using Ant to build and run the examples.

First, we have to tell the obfuscator the location of the classes to be obfuscated, and where the resulting obfuscated classes should be written. yGuard accepts JAR input and writes JAR output using the <inoutpair> tag:

```
<inoutpair
  in="./jmxbp.jar"
  out="./obfuscated/jmxbpObfuscated.jar"
/>
```

where jmxbp.jar contains the classes to be obfuscated, and the obfuscated classes will be written to a JAR file called jmxbpObfuscated.jar located in the obfuscated directory. Some obfuscators read JAR input, a relative directory to class files (where all classes located there and in subdirectories will be obfuscated), or a single class file.

Next, we tell the obfuscator what names we want to obfuscate. We can choose from any of our classes, fields, and methods by visibility, package pattern, name pattern, and so forth. yGuard allows configurable renaming of class, member variable (field), and method names as part of its <expose> tag. Anything you want exposed (i.e., not obfuscated) goes in this tag. There are many permutations of how

Apollo 17 Scientist-astronaut retrieving lunar samples. December 1972

ExtenXLS, the right tool for complex reporting™

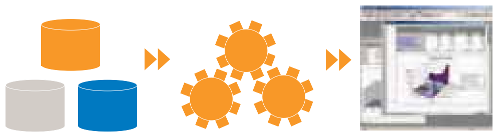
ExtenXLS 3, the 100% Java reporting toolkit

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

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

Download a free evaluation copy today: **extentech.com/jdjxls/**

EXTENXLS3 JAVA | XLS REPORTING TOOLKIT™



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

extenTECH™ **www.extentech.com**

renaming can occur, so it's impossible to show them all. But say, for example, that we want to expose only the public methods of our public classes. The yGuard configuration for this looks like:

```
<class classes="public"
  methods="public"/>
```

We can also choose to expose all public and protected methods, say, if our software is a library with classes intended to be subclassed. The yGuard configuration for this looks like:

```
<class classes="public"
  methods="public"/>
<class classes="public"
  methods="protected"/>
```

or we can selectively expose only certain methods of a class. We must take care to tell the obfuscator to expose the class name, so it may be referenced by name:

```
 <class name="jmxbp.common.Basic"/>
<method class="jmxbp.common.Basic"
  name="void reset()"/>
<method class="jmxbp.common.Basic"
  name="boolean isTraceOn()"/>
```

This configuration snippet will expose the Basic class and its reset() and isTraceOn() methods. Every other class and method will be obfuscated.

Finally, the obfuscator produces a log file (also referred to as a "map file") so that we can see the mapping between the original names of our classes, fields, and methods and their obfuscated names. This file can come in handy if, for example, you need to read a stack trace. The obfuscator may also provide a tool to automatically read in the map file, along with the obfuscated stack trace, and produce a meaningful stack trace. yGuard produces a map file, parts of which are shown in Listing 6.

Reconciling Open Source with Obfuscation

The open source software movement is the beginning of the commoditization of software. Since the early 20th century, as industrialized economies matured, the services sector boomed as fewer companies could compete against the dominant manufacturers, and more workers moved from factory jobs to the services industry. Because software isn't a product in the sense that, say, a length of PVC pipe is, the analogy between manufacturing and software development isn't airtight. At some point in the future manufacturing software systems may no longer be necessary. For example, there are certain standard sizes of pipe, and they pretty much all look alike. No one would consider custom manufacturing all of the pipes for a building onsite. Instead they are created by the manufacturer, ordered by the construction contractor, and shipped to the job site. But if you could copy a pipe the way you could copy a program, and the only "warehouse" you need for software is disk space, the manufacturer would be obsolete (or at the very least only a few would be extant). As more and different types of software move into the realm of open source, companies who may have traditionally manufactured and sold their software will reshape their business model around services such as support and customization.

However, as an industry, we are not there yet. Many companies manufacture software and maintain a competitive advantage by the way their software is written. These companies can use an obfuscator to help protect their software assets in a similar way that a wall protects a castle. No castle wall is impermeable, and no obfuscated code is completely safe from reverse-engineering attacks, but it does provide some level of defense. To continue the analogy, the better the obfuscator, the taller and stronger the walls.

What about an open source obfuscator like ProGuard? There seems to be a fundamental contradiction between the terms "open source" and "obfuscator." After all, the open source movement is all about sharing software for the benefit of the community. And the job of an obfuscator is to build a wall around software to protect it from reverse engineering. Or is it? In actuality, an obfuscator's job is to be the first line of defense in enforcing license agreements between the software company and those who would seek to gain an advantage via a reverse-engineering attack (i.e., "cheaters"). You might argue that reverse engineering a commercial product might be useful in solving problems, and, oh, by the way, avoid support costs to the vendor. However, I would argue that if you're reading this magazine, you're probably not the average developer, and wouldn't mind at all taking a little trip through the source code. Furthermore, I believe most reverse-engineering attacks are not aimed at avoiding support costs, or vendors who give away their products (along with source code) and who derive their revenue from the sale of support services and documentation would not be able to survive. But they do.

Things to Watch Out For

Make sure to properly expose classes, methods, or fields that are referenced by name (from your software or from the outside) using the Reflection API. If you don't, the names will not be found at runtime since they have been obfuscated. The sample application for this article makes heavy use of the Reflection API (see the DynamicMBeanFacade class) for building out the management interface of each class, so you'll see in the obfuscator configuration that I'm careful to preserve the appropriate method names accordingly.

Make sure to preserve native method names, so they can be linked to the correct native library.

Be careful when choosing what classes and methods to obfuscate. For example, if you're writing a library, you'll most likely want to keep public and protected methods and fields. Otherwise, your classes cannot be referenced by name. The sample application included with this article is standalone, so all of the classes with the exception of Controller are declared with public visibility. I chose to obfuscate all classes (except Controller and its main() method) since they are not to be called from the outside. When choosing which classes to rename, you'll also be forced to reexamine your design choices. Questions like "Why did I make that class public? It should be package private," or "That method is never invoked outside of itself, I should make it private," will come up, giving you the opportunity to improve your software.

Conclusion

While no software is safe from reverse engineering given enough time, patience, and persistence on the part of the reverse engineer, Java bytecode is especially susceptible. Because bytecode is architecture-neutral, a rich set of metadata is contained in the class file so that decompiling bytecode can very nearly yield the original Java source. A bytecode obfuscator, however, can rename packages, classes, member variables, and method names, making them meaningless. A sophisticated obfuscator can even alter control flow, making decompiled code even harder to read.

References

- Collberg, C.; Thomborson, C.; and Low D. "A Taxonomy of Obfuscating Transformations." Department of Computer Science, University of Auckland.
- Lindholm, T.; and Yellin F. (2002). *The Java Virtual Machine Specification, 2nd Edition*. Addison-Wesley.

Listing 1: The add() method of the Queue class after decompilation

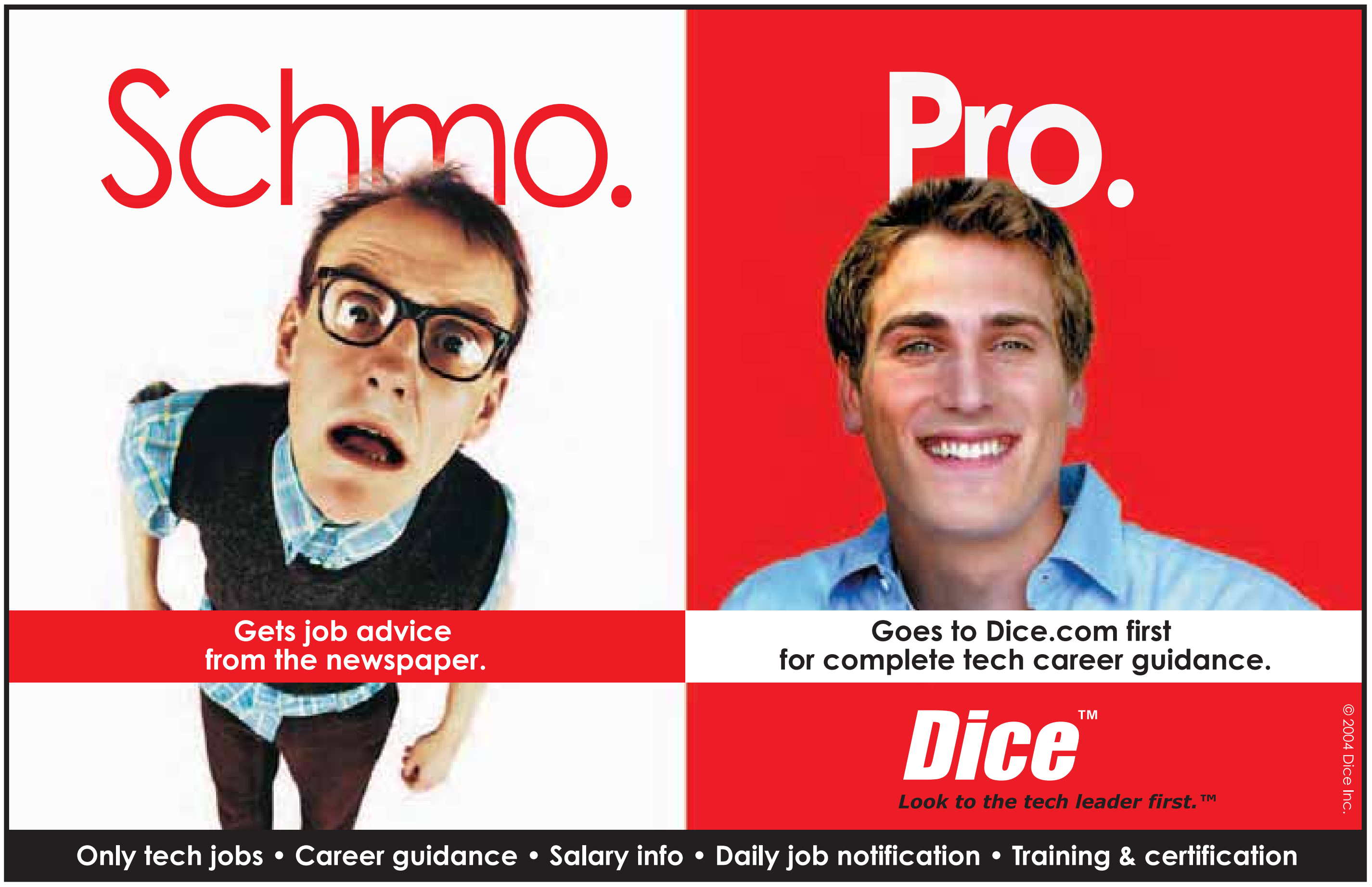
```
public synchronized void add
      (Object item) {
  long addWaitTime = 0;
  while (_suspended ||
         _tail == QUEUE_FULL) {
    long waitStart =
     System.currentTimeMillis();
    try {
      wait();
    } catch (InterruptedException e)
    {  }
    addWaitTime +=
     (System.currentTimeMillis() -
       waitStart);
  }
  _addWaitTime += addWaitTime;
  // add item to backing store
  _backingStore[_tail] = item;
  _queueEmpty = false;
  _tail++;
  if (_tail >= _backingStore.length)
    _tail = 0; // wrap
  if (_tail == _head) {
    _tail = -1; // special case:full
  }
  _queueFull =
    (_tail == -1) ? true : false;
  notifyAll();
}
```

Listing 2: The add() method of the Queue class

```
public synchronized void add
    (Object item) {
  long addWaitTime = 0L;
  while (_suspended || _tail == -1) {
    long waitStart =
       System.currentTimeMillis();
    try {
      this.wait();
    } catch (InterruptedException e){
      /* empty */
    }
    addWaitTime +=
      System.currentTimeMillis() -
        waitStart;
  }
  _addWaitTime += addWaitTime;
  _backingStore[_tail] = item;
  _queueEmpty = false;
  _tail++;
  if (_tail >= _backingStore.length)
    _tail = 0;
  if (_tail == _head)
    _tail = -1;
  _queueFull = _tail == -1;
  this.notifyAll();
}
```

Listing 3: The add() method of the Queue class after decompilation

```
public synchronized void add
    (Object object) {
  long l = 0L;
  while (_suspended || _tail == -1) {
    long l_2_ =
      System.currentTimeMillis();
    try {
      this.wait();
    } catch (InterruptedException e){
      /* empty */
    }
    l += System.currentTimeMillis() -
      l_2_;
  }
  _addWaitTime += l;
  _backingStore[_tail] = object;
  _queueEmpty = false;
  _tail++;
  if (_tail >= _backingStore.length)
  _tail = 0;
  if (_tail == _head)
    _tail = -1;
  _queueFull = _tail == -1;
  this.notifyAll();
}
```



Schmo.
Pro.
Gets job advice from the newspaper.
Goes to Dice.com first for complete tech career guidance.
Dice™
Look to the tech leader first.™
© 2004 Dice Inc.
Only tech jobs • Career guidance • Salary info • Daily job notification • Training & certification

Harness the **Power of Javadoc...**

...to track your team's code reviews

by James Scott

High-performing software teams use code reviews to improve the quality of their code and to keep team members informed of what's going on in the larger application. However, keeping track of which packages, classes, and methods have been reviewed can be a real headache, especially if a code review process is adopted for an existing codebase. Fortunately, Sun's Javadoc API documentation generator provides the basis for a simple but powerful tool that will help the team mark reviewed code and track the progress of reviews.

Most software teams recognize the virtues of code reviews. Though techniques vary in formality and thoroughness, all code reviews intend to find bugs as early in the development life cycle as possible. A developer who reads code that someone else wrote is likely to find issues that functional testing will not. Reviews also have a side benefit: developers get exposure to the techniques employed by their colleagues.

Going over the code is actually just the start of the code review process – someone has to keep track of which code has been reviewed and document the results of the reviews. This is not the typical developer's idea of a good time, but without this information the team will not be able to assess the effectiveness of their reviews. Spreadsheets, bug-tracking tools, and homegrown databases make adequate tools for tracking reviews, but they all suffer from a common weakness: they're separate from the code and therefore require an extra step after the review to keep the review statuses up-to-date. The ideal review-tracking tool would provide an easy way to mark the code as having been reviewed. As it happens, Sun provides just such a tool with the Java SDK: Javadoc.

James Scott is the software architect for Boston-based MedAptus, Inc. He has over 10 years of experience in designing and implementing mission-critical applications, and has worked with Java since 1999.

jscott@medaptus.com

Javadoc 101

Javadoc generates API documentation by parsing specially commented Java source code.

In the first versions of the JDK, Javadoc was hard coded to output HTML. Since JDK release 1.2, Javadoc has offered a public doclet API for producing custom output. Doclets control the format and content of Javadoc output. The standard doclet that ships with Javadoc generates the HTML-formatted documentation that you're probably familiar with, but third-party doclets are available to produce XML, PDF, and other formats. While doclets provide complete control over the output, sometimes they're overkill. As of J2SDK 1.3, Javadoc did not provide a means of making small alterations to the documentation content without having to rewrite the doclet.

In J2SDK 1.4, Sun introduced the taglet API for Javadoc. This lightweight API enabled the introduction of custom tags for use with any doclet, including the standard doclet. For the first time, developers had an easy means of customizing the standard HTML output. In the simplest cases, custom tags can be introduced on the command line, without any coding at all.

Below we'll see how to use both taglets and doclets to facilitate tracking of code reviews. The next section shows how to create a custom taglet that you can use to mark source code as reviewed. After that, we'll look at using a custom doclet to produce a management report.

Taglet, You're It

For a first pass at marking up source to reflect code reviews, let's define @review as a *standalone* tag, meaning that the Javadoc will look for it in the tag section that follows the main description. Taglets can also define custom *inline* tags, which can appear anywhere in the documentation comment. We want this tag to appear in type (class- or interface-level) documentation comments and method comments, since code is often reviewed in increments of methods or classes. The comment associated with the @review tag will specify the date of the review and who reviewed the code, so we'll have Javadoc output the string "Last Code Review:" as a title for the review comment.

The Easy Way: The -Tag Command-Line Option

When you don't need a lot of control over output formatting, Javadoc provides a very simple way of defining custom tags. The Standard doclet uses com.sun.tools.doclets.standard. tags. SimpleTaglet to format some of the standard Javadoc tags. If you don't need to modify SimpleTaglet's default output, you can implement a custom tag without writing any code at all. The -tag command-line option for the Javadoc tool allows you to specify the tag name, header, and valid locations for the tags, which Javadoc then uses to construct a Simple Taglet to process your tag. The format for this option is:

```
-tag tagname:Xaoptcmf:"taghead"
```

where tagname is the name of the tag (without the initial "@") and taghead is the header you want to appear in the output. Each letter in Xaoptcmf represents the set of elements where the tag should appear (see Table 1).

Executing the command:

```
Javadoc -tag review:tm "Last code review:"
```

on a source file with a class documentation comment containing an "@review" tag will produce output similar to that shown in Figure 1.

More Control: Extending SimpleTaglet

The previous example describes a fast way to implement custom tags but it does not provide any variation in output format. If you simply can't abide how SimpleTaglet formats your tags, you can create a class that extends SimpleTaglet and overrides its output methods. The main advantage to extending SimpleTaglet, instead of implementing the Taglet interface directly, is that SimpleTaglet takes care of the handful of trivial methods required by the taglet API that tell the doclet which elements the tag should appear in.

SimpleTaglet offers a three-argument constructor that accepts the same inputs as the -tag command-line option – hardly surprising, since SimpleTaglet underlies the command line. The first argument is the name of the tag, the second is the header to use in the output, and the third is a string that contains a single-character code for each element that the tag may appear in. Listing 1 shows a class that implements the review tag by extending SimpleTaglet. The static method register(Map tagletMap) is called during Javadoc setup and makes the doclet aware of the taglet's existence.

By default, SimpleTaglet puts the tag header in an HTML <DT> element and the tag comment goes in a corresponding <DD> element. Comments from multiple tags of the same type are concatenated into a comma-separated list in a single <DD> element. If this doesn't meet your needs, you need to override the toString(Tag) and toString (Tag[]) methods to provide the output you want. Javadoc calls one of these methods when it encounters a documentation element containing a tag handled by the taglet. Sun's documentation is a bit unclear on this point, but as of JDK 1.4.2, Javadoc always calls toString (Tag[]) for standalone tags, even if the documentation element contains only one tag of the appropriate type. Javadoc only invokes toString(Tag) when handling inline tags. Since SimpleTaglet is defined as a standalone tag, Javadoc will never invoke its toString(Tag) method. If you want significant differences in your processing of single and multiple tags – a table for multiple tags, for instance – check the array size and call toString (Tag) if the array contains only one tag.

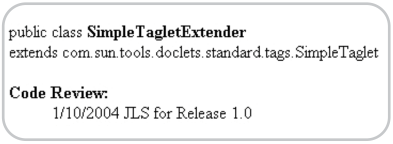
To compile the taglet, you'll need the J2SDK's tools.jar on your compile classpath, since it contains the taglet-related classes and interfaces. Once you have the class compiled, you can make Javadoc aware of the taglet with the following command:

Option	Meaning
X	Exclude this tag. This option overrides any other options that may be present.
a	Apply this tag in all elements. This option overrides any option except X.
o	Overview
p	Package
t	Type (class or interface)
c	Constructor
m	Method
f	Field

Table 1 Taglet inclusion command-line options

```
javadoc -tagletpath tagletdir -taglet
ReviewTaglet -sourcepath sourcedir
```

where tagletdir is the directory containing the taglet's class file and sourcedir is the directory holding the source code that you want to document.


public class **SimpleTagletExtender**
extends com.sun.tools.doclets.standard.tags.SimpleTaglet

Code Review:
1/10/2004 JLS for Release 1.0

Figure 1 Sample custom Taglet output

Implementing the Taglet Interface

For the vast majority of standalone custom tags, subclassing SimpleTaglet should provide the functionality you need with minimal implementation hassle. However, if you need an inline tag, you'll need to create a class that implements the com.sun.tools. doclets.Taglet interface. In addition to the toString methods described earlier, this interface defines methods to indicate whether the class describes a standalone or inline tag, several methods that tell Javadoc where to look for the tag, and one method to return the name of the taglet. In addition to the methods defined in the interface, the class will need to implement a static register(Map tagletMap) method to tell the doclet which tag name is associated with this taglet.

Introducing Doclets

So far we've explored how to add a custom tag to the standard doclet's output. This approach has its uses, but it's fairly limited. For instance, the taglet receives no information about the context in which it's invoked. Also, the standard doclet is designed to produce HTML, so our taglet has to follow suit if it's to be of any use. Showing the review status of a class in its API documentation is nice, but a summary report on the review status of a set of classes would be a far more useful management tool. Such a thing is beyond the capabilities of a taglet; we'll need a custom doclet instead.

Doclets are the first link in the documentation-generation chain and provide total control over the content and format of Javadoc's output. Javadoc invokes a doclet every time the tool is run. If no doclet is specified on the command line (using the -doclet option), Javadoc invokes the standard doclet. Except for the taglet API, there's no easy way to extend or modify the behavior of the standard doclet. To create our report, we'll have to implement our own doclet.

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.

Google℠

API Overview

Since doclets provide so much flexibility, it's not surprising that most of the API's functionality is devoted to describing the source code input to Javadoc. In fact, Javadoc doesn't even require a doclet to extend a specific class or implement an interface. To qualify as a doclet, a class must provide a static boolean start(RootDoc root) method. Though it's not required, Sun provides the abstract class.com.sun.javadoc. Doclet as a starting point. In addition to the start method, this class defines two methods for processing custom command-line options for the doclet. The real action is in RootDoc, which is the first in a set of interfaces from the com.sun.javadoc package that describe the source code that Javadoc was instructed to process.

RootDoc provides access to all of the packages and classes that were specified in Javadoc command-line options. It declares methods returning PackageDoc or ClassDoc types, which (as you can probably imagine) describe the structure and documentation comments of a package, class, or interface. PackageDoc's methods return ClassDocs for the classes in the package, filtered in various ways depending on which method is invoked. ClassDoc models the contents of a class, and uses ConstructorDoc, MethodDoc, and FieldDoc to describe its constituent elements. All of these interfaces extend Doc, which provides access to such things as tags, text of a documentation comment, and information about the position and type of the element currently being processed.

Code Review Report

Listing 2 contains the code for ReviewDoclet, which generates a summary report on @review tags in a set of classes. (Listings 2–4 can be downloaded from www.sys-con.com/java/sourcec.cfm.) Its start method uses PackageDoc.allClasses() to retrieve the list of classes that meet the filter criteria specified on the Javadoc command line. PackageDoc also provides an allClasses (boolean) method that allows the invoker to ignore the command-line filters. For each class in the package, we retrieve its ClassDoc and look for an instance of our custom @review tag. If one is present, we output its first sentence using the firstSentenceTags() method.

firstSentenceTags() returns an array of tags that represent the first sentence in the comment. If the first sentence contains any inline tags, the returned array contains one tag of kind = "Text" for each sentence fragment and one tag of the appropriate kind for each inline tag. This allows us to output a readable, plain-text version of the first sentence even if it contains markup.

Once ReviewDoclet is compiled, it can be invoked with the following command:

```
javadoc -docletpath docletdir -doclet
ReviewDoclet -sourcepath sourcedir
```

Custom Command-Line Options: Date Filtering on Reports

The Doclet API's flexibility carries over to the command line. Javadoc allows a doclet author to specify custom command-line options. Javadoc looks for two methods in the doclet class to parse and validate custom command-line options. The first, public static int optionLength (String option), is required and must be implemented to return the number of tokens for the option specified by the argument, or zero if the option is unrecognized. The second, public static Boolean validOptions (String[][], DocErrorReporter), is an optional method for integrity-checking of the arguments for custom options. The two-dimensional string array contains the names of all command-line options (not just custom options) and the arguments passed to each. The DocErrorReporter can be used by the method to print messages concerning validation errors.

This feature provides a useful extension to our code-review reports: date filtering. Some organizations specify that all code must be reviewed periodically. A report that shows which classes have been reviewed since a specified date will alert the reviewers to the parts of the codebase that need attention. By specifying a new custom tag called @reviewdate, we can use the doclet's custom command-line option to find out which classes have been reviewed since a specified date. Listing 3 shows the new methods and member variables for ReviewDoclet. The optionLength method is implemented to return 1 if a "-date" option is passed, and 0 otherwise. validOptions looks for a -date option and confirms that the associated argument contains a well-formed date. Since the doclet does not require a -date option, validOptions returns true if none is found.

Listing 4 contains a modified inner loop for the start method, which looks for a reviewdate tag and parses the text associated with the tag as a date. If the review date is after the date specified on the command line, the report simply outputs "OK." If the review date is absent, not parseable, or before the specified date, ReviewDoclet prints an appropriate message.

Conclusion: The Future of Javadoc

The biggest impending change related to Javadoc isn't in Javadoc itself, but rather in the extension of the concept of source code markup. The forthcoming J2SE SDK 1.5 includes an implementation of JSR 175, A Program Annotation Facility for the Java Programming Language. This JSR defines a syntax and API for source code annotations that can be read at compile time or runtime, instead of just by the documentation tool. The resulting metadata could be used for purposes ranging from source-code generation to compiler checking and code verification at runtime. JSR 175 is already somewhat controversial due to its "macro-y" nature and adoption of features seen in .NET, but it will doubtless have significant impact on the way large Java applications are written and constructed. Of course, Javadoc will also receive extensions so that the new annotations can be documented.

The code review documentation tools described here are a useful extension of Javadoc, but they represent only one extension of Javadoc's functionality. I encourage you to find ways to apply Javadoc to the problems facing your team. No longer will you be "working for Javadoc" – inserting the missing tags and malformed comments that Javadoc complains about. You now know how to make Javadoc work for you.

Resources

- *Javadoc Tool page:* http://java.sun.com/j2se/1.4.2/docs/tooldocs/javadoc/
- *Doclet overview:* http://java.sun.com/j2se/1.4.2/docs/tooldocs/javadoc/overview.html
- *Taglet overview:* http://java.sun.com/j2se/1.4.2/docs/tooldocs/javadoc/taglet/overview.html
- *Doclet.com (directory of third-party doclets and doclet information):* www.doclet.com
- *JSR 175 homepage:* www.jcp.org/en/jsr/detail?id=175

Listing 1

```
public class SimpleTagletExtender
    extends SimpleTaglet {

  public SimpleTagletExtender() {
    super("review","Code
Review:","tm");
  }

  public static void register(
        Map tgltMap) {
    SimpleTagletExtender tag =
        new SimpleTagletExtender();
    Taglet t = (Taglet) tgltMap.get(
            tag.getName());
    if (t != null) {

tgltMap.remove(tag.getName());
    }
    tgltMap.put(tag.getName(), tag);
  }

}
```

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

diagnosys

Are your Web applications open 24 hours? Join Sun's expert John VanSant and H&W for a Web seminar series April 6 and June 22 to learn how to make your J2EE applications run at top speed. Register today at: www.hwcs.com/wldj2.asp

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.

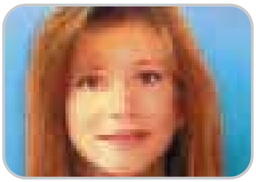
You need DiagnoSys.

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.

H&W
www.hwcs.com | 1.800.338.6692

Bridging the Enterprise Computing Gap For 25 Years



Java Trends

An interview with Amy Fowler

Interview by
Joe Ottinger

*Joe Ottinger, **JDJ**'s editor-in-chief, had the opportunity to talk with Amy Fowler, a senior staff engineer at Sun Microsystems and one of the founding members of the Java Swing GUI Toolkit, and discuss Swing, JSF, the JDNC, and some general trends in Java.*

JDJ: Can you tell us what your role is at Sun now?
Amy Fowler: Officially, I'm the technical lead of the Java Desktop Network Component (JDNC) project, which aims to simplify Java desktop client development for Web-enabled applications. Unofficially, I'm a rich-client agitator. I've been at Sun forever and have been an engineer on the J2SE client team since the days we called it the "JDK" and there were a total of eight packages. Most of that time I've spent on the Swing team, with a year-long tour of duty in J2EE as the JSF spec lead, trying to define a component model in the otherwise amorphous Web tier. So I have history, perspective, and a genuine passion for Java and user-interface software.

JDJ: What do you see Java being like in the future?
AF: Almost nine years ago Java hit a developer productivity sweet spot by providing a language that made just the right set of trade-offs between power and complexity. As the demands on software grow (more, better, faster, prettier), Java must target that sweet spot on a grander scale by making a more-capable platform easier to use. Tools are in there somewhere. If I may dream a bit, I hope Java will take advantage of the increasingly mind-blowing GPUs to deliver us from '80s-style user interfaces.

JDJ: As a JCP member (and former spec lead), how well do you think the JCP works? Are there any improvements that could be made.
AF: From my vantage point as a Sun engineer leading a JSR, it seems the JCP is a reasonable framework for involving the Java community in the evolution of Java standards. I think the process works best when a JSR is kicked off with a concrete proposal and working implementation as a starting point. Doing design by committee is like trying to run in the shallow end of a swimming pool.

One change we might need is an expiration on stale JSRs – the ones that get filed but never kicked off. If the leading company fails to take action, it shouldn't lock out other interested potential leaders/participants.

JDJ: Where do you see JDNC falling in the implementation hierarchy? Who are the primary users and developers, and when can we expect to see it in the marketplace?
AF: JDNC fits quite nicely on top of Swing and Java 2D. We've meticulously layered the implementation of JDNC so that developers can take advantage of it at many levels. If you're a Swing developer, you can use our Swing extensions package to get features like integrated table sorting/filtering/highlighting, data streaming, data binding, etc. The higher-level JDNC components wrap Swing and our extensions to provide a simpler API for common use scenarios. This API is targeted at Java developers who aren't necessarily Swing literate. Finally, the XML Schema layers on top to provide a declarative alternative to constructing the GUI, and this can be used by markup developers who may or may not know Java.

My June '03 white paper (www.javadesktop.org/articles/JDNC) focused mostly on the XML layer, but we've spent most of this last year developing the underlying Java APIs, whose simplicity should be reflected in the XML. We don't view Java coding and markup as mutually exclusive; in fact, we expect a majority of applications will do both; JDNC will encourage a sensible division of labor between the two. We hope to get the bits on javadesktop.org before JavaOne to encourage you all to come talk to us.

JDJ: How does JDNC compare with JSF? JDNC's stated goals include less round-trip data, while JSF is likely to mandate more round-trip data – is there a middle ground from Sun somewhere?
AF: Okay, I won't hide my bias. Excessive round-trips and associated page refreshes are an inherent limitation of HTML applications that resulted from jamming a square peg in a round hole – building GUI clients out of a document-viewing substrate. Nonetheless, HTML applications have become a major force in our daily lives and many have gotten quite good (Amazon paved the way). JSF is all about making those Web apps easier to build, which is a very good thing once you understand that doing it well is tricky indeed.

To borrow a term from Dr. Jakob Nielsen (www.useit.com), I think browser clients are well suited for the "ephemeral" use case where users perform limited tasks of a sequential nature. Rich, interactive clients deliver the tempo required by applications that are used intensively. Unfortunately, the user experience is usually not the driving factor in making the decision between the two.

But your question of a "middle ground" comes up a lot and I've been squinting to find it for some time. One potential integration idea is to build some JSF renderers that emit JDNC components as applets in key areas of the page to dial up the view interactivity, like providing a tabular data view that can be scrolled, sorted, filtered, and rearranged without page refreshes. Where this breaks down is when the user triggers an event that is routed back to the server and causes JSF to rerender the page. The applet (and data) must then be reloaded. Bummer. But software has amazing pliability, so I suspect we can solve these problems if we decide they need to be solved.

JDJ: Why has JSF focused solely on Web content? To me, being able to describe a working interface for all client models would be a huge benefit, as I typically develop Web applications but can see a valid use for some Web apps being desktop apps. So far, SwingML, XUL, thinlets, JSF...all are trying to do the same thing in different domains, which seems to beg for optimization.

AF: This is the holy grail of GUI development – the ability to define the UI once and have it magically map to the broad range of scenarios that face users today from small devices to browsers, to desktop apps, etc. The following is my personal assessment of why this won't work.

First, each scenario has distinctly different operational characteristics, such as screen-size, input device, memory capacity, network latency break points, etc., that require thoughtful reconfiguration of the user experience. For example, there's nothing more annoying than an HTML application that tries to act like a rich client – having the page refresh when the user makes a selection from a dropdown choice is an abomination of modern software. A decent client should leverage the network without obfuscating it.

Second, we are by nature visual creatures and are ultimately obsessed with having precise control over the bits on the screen. This almost always requires tinkering, which is tailored for the specific client technology. We'd never be satisfied with visuals that are generated or limited by the least common denominator.

Ranting aside, there is definitely room for significant sharing of some of the nonvisual bits, such as data models, validation, business logic, and application behavior. JSR 227 (data binding) promises the requisite data-binding scheme to support this and we're counting on Oracle to see this through [in the meantime, data binding will be a major feature of JDNC].

JDJ: Since I've brought up Swing...in the JDNC document, you mention some things in Swing as being nontrivial. Given that most Swing interfaces certainly don't seem to leverage what Swing can do (or don't leverage it well), do you see Swing improving in ways that would change that? Do you think Sun will ever revamp Swing's documentation to the point where the average Swing programmer no longer shrugs at "Swing's" poor performance? (I quote "Swing" there because it's my opinion that programmer education is the problem, not Swing.)

AF: Let's nail this right here. Due to tremendous efforts by the Java 2D, AWT, and Swing teams, and the kindness of Moore's Law, Swing performance is no longer a valid excuse for not using Java on the client. Thanks to Chet Haase and the Java 2D team, 2D will continue its trend to rely more on the graphics hardware acceleration. Swing's lead, Scott Violet, did work in 1.5 to reduce Swing's memory footprint somewhat. There are a bunch of creative discussions going on with members of the VM team on improving startup. I use a number of Swing applications daily on a three year old laptop and performance is just not a problem.

Now, one consistent and fair criticism is that it's harder than it should be to write a performant Swing app, which leads right to the issue currently burning holes in my laptop. How do we make Swing more tenable to a broader range of developers? We have to make it easier to approach and ensure developers can achieve better results with less effort. Swing is indeed broad and fine grained. This was intentional. We didn't want to limit the kinds of GUIs that could be developed in Java.

It took us awhile to realize that while we get twisted pleasure out of writing the threading code to load data into models asynchronously, 99% of the world does not. It's quite reasonable to simplify these common or difficult tasks by layering APIs (like JDNC), but we also need better tools and documentation.

One theory I'm still trying to prove is that our monstrous Javadoc is an issue. There are indeed more methods on JButton than there ought to be, yet we should be able to emphasize more clearly in tools and documentation the two or three that developers really need. Javadoc is the reference manual, yet I suspect it's also relied upon as a programmer's guide. I definitely use it that way – it annoys me when I also have to download a pdf spec to figure out how to use an API.

JDJ: As a corollary to the previous question, are the new technologies coming out of Sun prioritizing documentation and education? How much are they doing so, or are they focusing on the technology alone?

AF: Our priorities are clear: simplification and ease of development. Documentation and education are obviously large components of that, but it feels like we could do more of the latter. Our competition is very good at that.

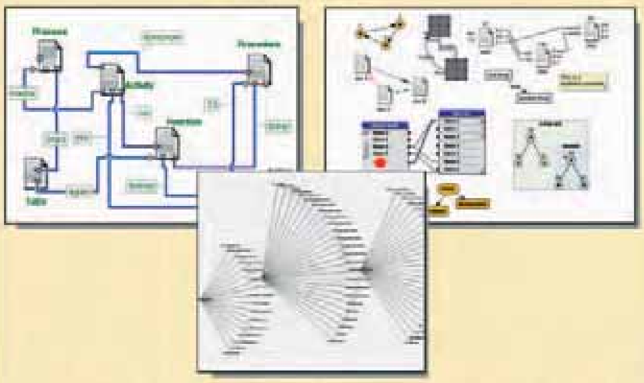
JDJ: Will JDNC follow the JavaBeans component model?

AF: Yes. As much as possible, we're trying to make Swing functionality accessible through the beans patterns. For example, in our high-level table component, you'll be able to configure column rendering (colors, alignment, etc.) as bean properties without having to understand Swing's underlying CellRenderer mechanism.

JDJ: JavaBeans are used as a server-side component model, whereas they began life as a client-side tools API (with things like property editors and customizers). Are there any moves to embrace more server-side declaratives, or move toward the metadata model coming in 1.5?

AF: I believe that JSF has some facilities for managing beans in the Web tier. I recommend checking out the specification for details.

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*

Northwoods SOFTWARE CORPORATION

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

JDJ: As a corollary, how do the changes coming in 1.5 affect current designs?

AF: Version 1.5 is the most exciting Java release in some time. It's my personal misfortune that the current version of JDNC must run on 1.4.2, leaving me little time to play with 1.5 features directly. You all probably have more experience than I do in using 1.5's generics and autoboxing at this point! Unfair, but true.

In Swing, the new "Synth" look-and-feel will enable a whole new way to customize the look of an application without touching a line of code. This was part of the original vision of Swing – it just took us seven years to get to it. With 1.5, developers can skin applications without knowing anything about the pluggable look-and-feel (plaf) packages because the visual elements of the GUI can be configured in XML. Now we just need the tool that lets graphics designers, rather than engineers, create the skins.

For JDNC ,we're looking forward to leveraging Swing's new table printing, Java WebStart enhancements, the Xerces parser, and WebRowSet.

JDJ: How do you see Swing and SWT interoperating in the future?

AF: The AWT team made some changes in 1.5 to better support the hosting of Swing components within SWT (and Eclipse). It would be nice if they could be mixed and matched more freely; however, my understanding of the respective architectures is that this would be horrendously difficult without incompatible changes on one or both sides.

In light of Java's real and very mean competition (.NET and what's coming behind it), the last thing Java needs is a toolkit war. I've been there before – it's a complete waste of time. Our competitor gets great joy out of watching the infighting within the Unix community. Let's not go there with Java.

"Our priorities are clear: simplification and ease of development"

JDJ: With the LayoutManager2 interface the constraint object is passed as an argument to the container's add method. Why is the constraint not a property of java.awt.Component? Right now the only way to change a constraint is to add and remove the component.

AF: This does seem like an oversight that needs to be fixed. I'll plead the case with the AWT team.

JDJ: Are there any plans to make AWT and Swing interoperate better? If not, are there any plans to add Tree, Table, and TabFolder to AWT because this would go a long way toward satisfying the needs of people who want Swing's power coupled with native components?

AF: AWT and Swing interoperate fairly well as they share the same component model. We do have plans to make the heavyweight/lightweight clipping issue go away, hopefully, in the release subsequent to 1.5. We don't, however, currently have plans to build peer'd components for Tree, Table, etc. Look-and-feel fidelity is a big priority for us and we'll always be investigating ways to improve it, but likely in the context of the Swing components. However, the underlying platforms do evolve and we'll periodically reevaluate the best way to achieve fidelity.

(Q&A with Amy Fowler continued at www.sys-con.com/java.)

BEST PRACTICES FOR MANAGING YOUR J2EE APPLICATION PERFORMANCE

Quest Software understands the unique challenges Java developers face today, and has responded with tool sets that help identify root causes of J2EE performance issues, isolate lines of code that cause bottlenecks, automate the performance process and ensure third-party J2EE application performance. Whether a company maintains a rigidly defined development process or responds to performance and reliability issues as they come up, Java developers know that performance testing has never been more important. Developers need to ensure that the project they have invested their time and energy in is successful - not only in their eyes, but also in the eyes of management and, most importantly, the end user.

To ensure that your J2EE applications are running at peak performance, consider joining the thousands of companies worldwide who depend on Quest Software to detect, diagnose and resolve J2EE performance issues across the development lifecycle.

Your Challenge: Quickly Identifying the Root Cause of J2EE Performance Issues

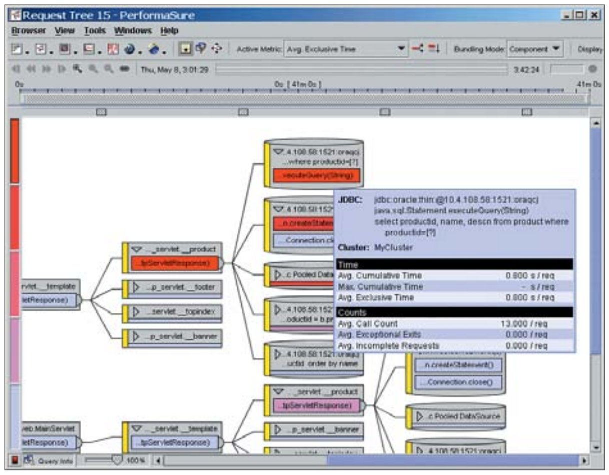
J2EE and portal development is complex. The code developers write interacts with the assorted frameworks, possible third party components/ libraries and leverages the resources of various environments. All applications have key transactions or use cases that must perform optimally. To quickly narrow down investigation, developers need to look for where time is being spent when these key transactions run. For example, is time spent in the Web tier, in calls to the database or in your EJB or portal code?

Find out with Quest's PerformaSure™, a powerful tool for diagnosing J2EE problems that allows development teams to accurately and quickly locate and resolve the performance bottleneck.

Your Challenge: Isolating the Line(s) of Java Code Responsible for Your Performance Bottleneck

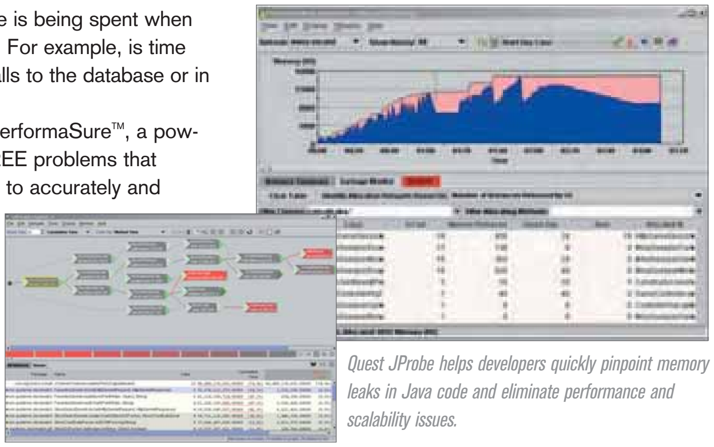
In the end, Java developers are responsible for changing Java code to ensure that any identified performance bottlenecks or memory leaks are resolved. If these problems are located early in the development lifecycle, then the company and the developers will save valuable time and resources compared with a more reactive approach. Developers need tools to help them diagnose and understand performance and memory issues at a line of code level.

Developers can use Quest Software's JProbe® Profiler and JProbe Memory Debugger to understand how their code uses the resources available. By understanding which lines of code take the most time to execute, which objects are persisting in memory after garbage collections, which methods are called the most and by whom, developers can then make the necessary changes to the application code to ensure performance bottlenecks are eliminated.


Request Tree 15 - PerformaSure
Browser View Tools Windows Help
Active Metric: Avg. Exclusive Time
Bundling Mode: Component
Thu, May 8, 3:01:29
3:42:24
0s [41m 0s]
41m 0s
JDBC: jdbc:oracle:thin:@10.4.108.58:1521:oraqcj
java.sql.Statement.executeQuery(String)
select productid, name, descn from product where productid=[?]
Cluster: MyCluster
Time
Avg. Cumulative Time 0.800 s / req
Max. Cumulative Time - s / req
Avg. Exclusive Time 0.800 s / req
Counts
Avg. Call Count 13.000 / req
Avg. Exceptional Exits 0.000 / req
Avg. Incomplete Requests 0.000 / req
_servlet_product
_tpServletResponse)
_p_servlet_footer
_servlet_topindex
_p_servlet_banner
_servlet_template
c.Pooled DataSource
n.createStatement()
Connection.close()
Query Info
100%
PerformaSure's Request Tree View highlights the most time-consuming component, method or JDBC calls with color-coded hotspots.

Your Challenge: Automating the Performance Tuning Process

Quest Software understands that there are many demands on developer resources, which is why time saving features are an integral part of JProbe. Using JProbe's trigger facility, ANT task integration, snapshot differencing and command line interface, developers can automate the process of collecting performance information and integrate with their build and test systems. By automating the collection of this valuable diagnostic information, developers free up time for more crucial responsibilities.


Quest JProbe helps developers quickly pinpoint memory leaks in Java code and eliminate performance and scalability issues.

Your Challenge: Ensure Performance of J2EE Applications Developed Offshore

A growing challenge for companies is managing the quality and performance of applications that have been developed by third-party organizations. Even without a clear understanding of the code structure and the exact testing that has been performed on the code prior to reaching an organization, development, testing and QA teams are still ultimately responsible for the success of the application. By using a combination of JProbe and PerformaSure along with other functional testing tools, developers can ensure that their applications perform at an optimal level. If they do happen to find performance issues, they can quickly identify the root cause and provide specifics to the offsite development team to ensure the issue is resolved quickly and correctly.

All organizations need to move faster and more efficiently each day in order to win business. Java developers are being asked to take on responsibility for more and more – and given less time to deal with it all. As a developer, whether you need tools that will allow you to treat performance and reliability issues more proactively, or merely react faster to incoming emergencies, we can help. Let Quest Software show you how our market-driven tools and experience can help you to quickly find performance problems throughout your development lifecycle.

You know performance is important, let us show you how easy taking control of it can be. Visit us at: www.quest.com/jdj_advertorial and learn more about ensuring better performing J2EE applications with Quest Software's JProbe and PerformaSure.

QUEST SOFTWARE® www.quest.com

Joe Winchester
Desktop Java Editor

Less > More

Among geneticists there is an ongoing argument about which species is superior: humans or bacteria. Both are the end product of millions of years of evolutionary refinement; they just took separate routes on the road to survival. Humans represent the pinnacle of animal development, possessing feature upon feature such as communication and the ability to alter the environment to suit their needs, while bacteria took the opposite tack and remained single-celled organisms that adapt to fit their environment. Large animal species tend to be very vulnerable to environmental change as each ice age or asteroid collision tends to result in a mass extinction. The small and nimble bacteria, however, can weather all storms that the planet throws at it; the irony is that bacteria predate us and will outlive us.

I'll segue this into application user interfaces. When building software tools it's often good to put yourself in the mindset of the users who don't have your program available to them. Without the functionality you are about to add, what would they do instead? It should always be possible to not use a tool; without a particular tool a developer will just fall back to editing raw files in a text editor and running scripts to do builds or database updates. The job of a program is to take the raw APIs and package them in a form that make sense at a higher level, adding features such as validation, persistence support, or transactional integrity. Once these tasks have been done, a user interface can be added so that the execution of the tasks is easy and intuitive to the user.

“One thing evolution should teach us is that sometimes more is not necessarily better and that refined simplicity stands the test of time”

Before immediately writing a lot of GUI code, if the user without the tool was going to edit raw files and run scripts, why not work with rather than replace this environment? Some folks swear by their favorite text editors like VI, emacs, or SlickEdit, priding themselves on their ability to customize and configure macros and shell scripts. I've seen developers who when running into a particularly nasty problem crank up a command prompt and start grepping log and trace files to find out what is really going on. I used to always view it as a failure of the high-level tools if they weren't able to fully replace the need for developers to have to inspect raw files, but now I think it's just a different application species. Text editors came before the GUI and will outlive the fads of client/server to Web client to rich client and back. For me, as a Java developer, the features I appreciate most are those that help with writing code by coloring source, highlighting errors as I make them, or offering code completion choices. When the electronic keyboard was created, the piano keys were kept as they had been for hundreds of years with features added for sampling and mixing the sound. At what point does a computer program reach nirvana for its user interface?

If the most long-lived applications are those that fit themselves around the existing interface, adding and augmenting what a user would do if the program weren't there, how does this relate to business applications? Before computers there was lots of paper and filing cabinets. Databases replaced the filing cabinets; however, apart from being a good way to archive and store information, databases didn't fundamentally fit into the way companies run. In any organization information is collated, files are dealt with or passed on for attention, and the data flows around the company. I'm not sure what the best user interface is to deal with the real world, but it is one that mirrors a world without the computer, where paper trails are kept, phone calls made, and meetings take place. I even question whether or not database records should be so formatted into rigid fixed columns that, while they help to collect data, they constrain what information can be held and arguably are just there so that fixed format reporting can occur.

At some point in the future the kind of applications we use will look horribly dated. Something far better will come along that completely replaces the keyboard and monitor. Jakob Neilsen has a nice list of books that talk about the user interface of the future (www.useit.com/books/future books.html), discussing ideas ranging from three-dimensional globes to visualize weather, to a virtual reality markup language. Whatever it will look like, one thing that evolution should teach us is that sometimes more is not necessarily better and that refined simplicity stands the test of time, and the best computer interfaces augment what the user would do if the application wasn't there.

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

joewinchester@sys-con.com

DEPLOY SOA. NOW.

Service-Oriented Architecture. It's the buzz in the world of enterprise software development and integration: Web services and service-oriented architecture (SOA). And at BEA, we're all over it.

SOA—and the growing use of discrete, self-contained business services—has become widely recognized as the most relevant strategic direction for enterprise computing. And increasingly, the BEA WebLogic™ development environment is recognized as the premier platform on which to implement it. Right now.

As a ground shaking new architecture, SOA is making waves throughout many organizations. Come to BEA eWorld 2004 to see how easily you can deliver SOA in your organization NOW!

BEA eworld 2004

REGISTER NOW!

www.bea.com/eworld

MOSCONE WEST CONVENTION CENTER SAN FRANCISCO, CA MAY 24-27, 2004

Thanks to our generous sponsors

Diamond Sponsors

hp invent intel

Platinum Sponsors

accenture High performance. Delivered.

COMPUWARE

bea

DESKTOP | CORE | ENTERPRISE | HOME

ERROR RECOVERY

Facilitate problem resolutions in both Web and desktop systems

by Todd C. Brett

Java developers have access to a variety of error detection tools, such as try/catch blocks and variants of the Exception class, but few mechanisms provide a structured, interactive recovery. As a result, users are frequently faced with frustrating application instability that can be difficult to resolve. Fortunately, an error recovery framework can be easily created and integrated into almost any desktop or Java server application.

Progressing from Error Detection

A typical application begins the error handling process by catching an Exception object bearing some kind of problem description, whether by virtue of the class type or attribute data. The system might also perform error checking and data validation on method return values for situations in which corrupt data will not cause a crash outright. Ideally, the application should be able to analyze each situation and take appropriate measures to repair the environment so that the task can be successfully completed. What typically happens instead is that the afflicted task is aborted, perhaps with a report or log of some kind.

```
try
{
  StringBuffer helloWorld = null;
  JOptionPane.showMessageDialog( null,
                                 helloWorld.toString() );
}
catch ( Exception error )
{
  JOptionPane.showMessageDialog( null, "Oops!" );
  System.exit( 1 );
}
```

The problem is that providing recovery logic for each error situation can be a time-consuming task whose solutions must be distributed throughout a system's logic. Each application is unique, so preparing a general framework to handle error situations is a challenging task that many library and API development teams prefer to leave to the application developers.

The solution in these situations is for development groups to create their own framework that allows common logic to be reused within a context that allows for interactive system guidance. The following roles facilitate this progression from error detection to error resolution:

- Problem description
- Suggested solution
- Resolution manager
- Resolution listener
- Resolution validator

Todd Brett has over 10 years of experience architecting and implementing well-behaved, multiplatform applications in a variety of languages. He holds an MS with an emphasis in object-oriented technologies and technology management from Regis University in Denver.

toddbrett@insight.rr.com

Describing the Problem and Its Solution

The error description is the best place to begin when designing a recovery framework. Systems have the option of specifying a problem component that either serves as the base class for application exceptions, or is distinct with error information as a constructor parameter. The latter strategy works best with applications that are tightly bound to Sun's libraries and virtual machine because the exception architecture remains consistent. Regardless, the component must describe the problem in terms that both the user and the system can understand. The problem can be represented by a variety of perspectives that range in complexity – from a simple text string to a domain expert facility with access to a wide variety of resolution and logging mechanisms.

```
public interface IProblem
{
  public void setError( Throwable p_error );
  public String getErrorDescription();
  ...
}
```

The logic that discovered the error will typically create the problem object and work to resolve it, but a collection known to the application should track the object for later analysis to ensure persistence beyond the current operation.

```
catch ( Exception error )
{
  // Log the issue for later reference.
  problemCollection.add( new Problem( error ) );
  ...
  // Work to resolve issue.
}
```

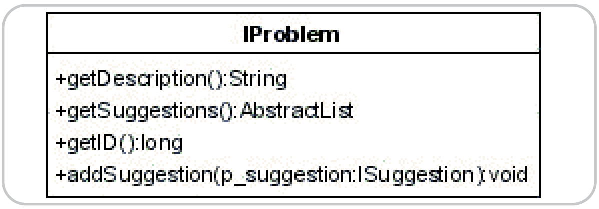

IProblem
+getDescription():String
+getSuggestions():AbstractList
+getID():long
+addSuggestion(p_suggestion:ISuggestion):void

Figure 1 A problem description component

June 28–
July 1, 2004

Moscone Center
San Francisco, CA

Join James Gosling, the father of the Java™ programming language

Java™

Everywhere starts here

Java™ technology is everywhere, improving the digital experience for everyone. It all starts at the JavaOne℠ conference, your source for cutting-edge knowledge and proven solutions. Discover from the experts how to deploy Web services and connect the world securely; you'll learn to code simpler and faster, and bring higher efficiency and profitability to your business.

The JavaOne conference offers hundreds of in-depth technical sessions in:

- Topic 1—The Foundations: Core J2SE™ Technologies
- Topic 2—Core Enterprise Technologies
- Topic 3—Java™ Technology on the Desktop
- Topic 4—Java Technology for the Web
- Topic 5—Java Technology for Mobility
- Topic 6—Dissecting the Implementation: Solutions
- Topic 7—Intriguing and Unexpected: "New and Cool"

JavaOne℠

Sun's 2004 Worldwide Java Developer Conference℠

Save $200! Register by May 31, 2004, and receive the Early-Bird price for the full Conference package. Registration code: ADARZKND

Register at
java.sun.com/javaone/sf

Sponsored by

Sun microsystems

Produced by

MediaLive™ INTERNATIONAL

Figure 1 illustrates a problem description component that offers a reasonable amount of error and tracking detail.

The error handling logic can also instantiate solutions suggested for the problem because of the failing operation's intimate familiarity with the process and the expected environment state. Not all solutions can be automated (such as the need for a missing disk), so the solution role is best broken into two classes: a suggestion component to provide resolution instructions and a solution component that automates the resolution for execution by the system on behalf of the user. Suggestion objects are directly associated with a problem in order to provide system access regardless of where the error is to be resolved, and each suggestion in turn can be associated with a solution object if the system can act on behalf of a user for the resolution attempt.

```
public interface ISuggestion
{
  // Call this if the suggestion can be automated.
  public void setSolution( ISolution p_automatedSolution );
  ...
}
```

Designers should keep in mind alternate description or instruction strategies that offer additional graphical or animated illustrations to support other language groups or people with varying disabilities. Figure 2 illustrates the relationship between potential solutions and the problem description.

There's little reason for the role functionality of IProblem and ISuggestion to vary so concrete convenience classes can be provided for these two components. ISolution's role will almost always be custom to a specific application or library so its implementation is left to the development team.

Simultaneous attempts to solve a problem should be avoided because the ideal benefits of the structured error recovery include consistency and predictability. As such, synchronization techniques need to be used with all of a problem's methods, which has the added benefit of allowing the internal ISuggestion collection to use fast data structures.

ISolution serves as a facade for all of the operations involved in a single attempt to recover from an error. Therefore, Suggestion only needs to contain one reference to an associated ISolution, and a value of null can represent a user-driven solution. This one-to-one relationship ensures that the user retains a clear understanding of the environmental and data consequences of any error resolution attempt, which in turn increases the user's confidence in the integrity of the system he or she is using.

A simple Suggestion class will often provide only a text description of the potential solution's steps and consequences, but architects should also consider the use of animations to clearly communicate what will occur with the suggestion's selection. Wizards can be used in automated solutions to collect custom data values that will affect the nature or severity of the consequence, such as how far to adjust steam pressure in overloaded industrial equipment.

Guiding the Resolution Process

The selection of a suggested solution is best provided through a component that can consistently orchestrate the resolution process. The Mediator pattern provides an excellent template for the problem-solver component by logically connecting the problem and its solutions to the needs and insight of the application. Figure 3 illustrates the IProblemSolver interface and its relation to IProblem.

Concrete implementations of IProblemSolver accept a specific problem, execute solutions as appropriate, and then check to determine if any given solution fixed the problem. Problem solvers can either act independently, query the user for guidance, or a combination of both as appropriate for user accessibility and the failing operation.

```
...
IProblemSolver solver =
  new AutomatedProblemSolver( someProblem );
boolean resolved = solver.solveProblem();
```

Automated problem solvers that use only ISolution objects are often best for server-side systems and client tasks that are trivial in their resolution. AutomatedProblemSolver is an example component that should be provided by the framework for just such a purpose. AutomatedProblemSolver looks through a given Problem instance for valid ISolution instances and organizes their execution according to a success likelihood indicator supplied by the associated Suggestion. Architects need to be careful when using automated problem solvers, however, because the user has no control over the outcome other than what executed solutions might provide. In fact, the solutions provided will often require expert knowledge about the system and environment, and the application may need to intercede between resolution attempts by the problem solver to reset any environment variables left in an inappropriate state from a failed attempt.

Alternatively, user-driven problem solvers take advantage of all provided ISuggestion objects by presenting to the user a subset of those most likely to succeed. The primary benefit of an interactive problem solver is the control felt by the user, but situations can also arise in which the user is essential to the task at hand through intimate knowledge of the environment or access to resources not programmatically accessible. The recovery framework can provide such a com-

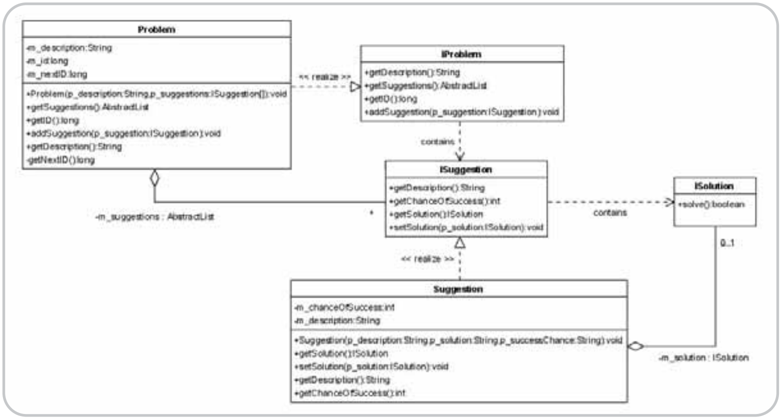

Problem
-m_description:String
-m_id:long
-m_nextID:long
+Problem(p_description:String,p_suggestions:ISuggestion[]):void
+getSuggestions():AbstractList
+getID():long
+addSuggestion(p_suggestion:ISuggestion):void
+getDescription():String
-getNextID():long
<< realize >>
IProblem
+getDescription():String
+getSuggestions():AbstractList
+getID():long
+addSuggestion(p_suggestion:ISuggestion):void
contains
ISuggestion
+getDescription():String
+getChanceOfSuccess():int
+getSolution():ISolution
+setSolution(p_solution:ISolution):void
-m_suggestions : AbstractList
*
contains
ISolution
+solve():boolean
0..1
<< realize >>
Suggestion
-m_chanceOfSuccess:int
-m_description:String
+Suggestion(p_description:String,p_solution:String,p_successChance:String):void
+getSolution():ISolution
+setSolution(p_solution:ISolution):void
+getDescription():String
+getChanceOfSuccess():int
-m_solution : ISolution

Figure 2 The relationship between potential solutions and the problem description

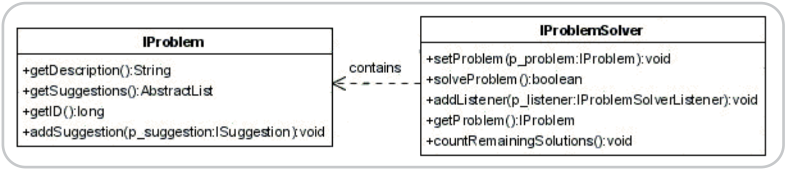

IProblem
+getDescription():String
+getSuggestions():AbstractList
+getID():long
+addSuggestion(p_suggestion:ISuggestion):void
contains
IProblemSolver
+setProblem(p_problem:IProblem):void
+solveProblem():boolean
+addListener(p_listener:IProblemSolverListener):void
+getProblem():IProblem
+countRemainingSolutions():void

Figure 3 The IProblemSolver interface

LINUXWORLD CONFERENCE & EXPO

CONFERENCE: August 2 – 5, 2004 EXPO: August 3 – 5, 2004

MOSCONE CENTER • SAN FRANCISCO, CA

Where OPEN MINDS Meet

LinuxWorld Conference & Expo is the #1 venue for decision-makers and influencers to discover real business solutions for real business problems, learn how Linux is accelerating as a total enterprise solution, and understand how it can be applied effectively to save their companies time and money.

Attend LinuxWorld and...

- Realize Linux as more than an operating system, but as **a world of applications**
- Evaluate the **latest applications and innovations** from leading open source companies
- Explore interoperability issues and opportunities in **open source and proprietary environments**
- Stay on the cusp of **emerging technologies** and the acceleration of open source adoption in enterprise computing
- Review the latest open source initiatives, their deployment and successes to help **make informed decisions** for your company

WWW.LINUXWORLDEXPO.COM

Register Online With Priority Code: D1401

CORNERSTONE SPONSOR

ORACLE

PLATINUM SPONSORS

DELL IBM intel Novell

IDG WORLD EXPO

"Simultaneous attempts to solve a problem should be avoided because the ideal benefits of the structured error recovery include consistency and predictability"

ponent, GuidedProblemSolver, that uses Swing or a similar technology to present the user with an error resolution dialog containing the top three suggestions. The resolution is iteratively attempted by users through the selection of a suggestion and the execution of its instructions. Automated solutions are supported through the association between ISuggestion and ISolution components, but the suggestion should clearly indicate the automated nature. A cancel button should always be provided by the dialog to allow the user the opportunity to cancel the task instead of resolving it (for example, a disk that is not on hand might be needed). The application should still communicate with the problem solver after each resolution attempt to ensure that the environment is in an appropriate state for the next attempt.

Just as with suggestions and solutions, several variations of the problem solver can be introduced to applications. For example, a wizard could query users on their preference for resolution with regard to the existing state of the operation, much in the same way some applications present users with a series of questions to determine which help document would best meet the user's needs. Alternatively, a Web system might use a problem solver to track the session and generate HTML forms that simulate an interactive dialog. As can be imagined, problem-solver components can become complex applets in their own right, and so require special attention from the technical and business experts.

Application Participation

The confirmation or rejection of an attempted solution is an important aspect of error resolution. A problem solver could be used for this task, but a role-based component better allows the application's seamless integration into the process without affecting the more general nature of the error recovery framework. Figure 4 illustrates the relationship between a problem solver and a resolution validator.

A concrete implementation of IResolutionValidator will be, by necessity, a domain expert component that is specific to the operation affected by the error. The resolution process can be complex because the observing component must intimately understand what a stable operation looks like from execution, data integrity, and environment perspectives. Validation logic must consider whether all three aspects of the operation have been repaired by an attempted solution; if not, the validator must consider whether additional attempts should be made, or if the operation will at least work well enough to allow the data to be saved in anticipation of a new session. For example, word processors often create backup files that can be reopened once the application has been restarted, and many systems can behave in a similar fashion for calculational tasks. However, it's important to understand that validators are not responsible for performing any repairs upon the system, but rather must only communicate to the problem solver whether or not the given problem remains an issue. Environment adjustments and data rescue are performed by solutions or additional problem-solver observers.

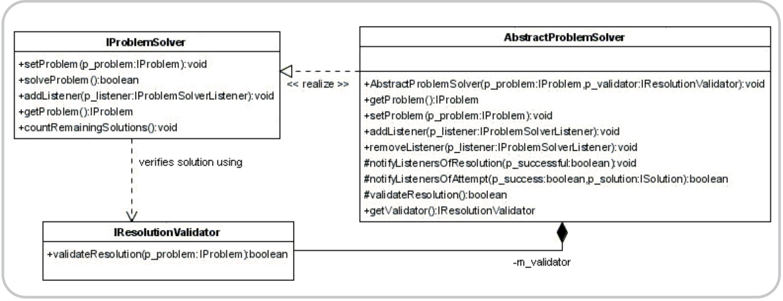

IProblemSolver
+setProblem(p_problem:IProblem):void
+solveProblem():boolean
+addListener(p_listener:IProblemSolverListener):void
+getProblem():IProblem
+countRemainingSolutions():void
<< realize >>
AbstractProblemSolver
+AbstractProblemSolver(p_problem:IProblem,p_validator:IResolutionValidator):void
+getProblem():IProblem
+setProblem(p_problem:IProblem):void
+addListener(p_listener:IProblemSolverListener):void
+removeListener(p_listener:IProblemSolverListener):void
#notifyListenersOfResolution(p_successful:boolean):void
#notifyListenersOfAttempt(p_success:boolean,p_solution:ISolution):boolean
#validateResolution():boolean
+getValidator():IResolutionValidator
verifies solution using
IResolutionValidator
+validateResolution(p_problem:IProblem):boolean
-m_validator

Figure 4 The relationship between a problem solver and a problem-solver listener

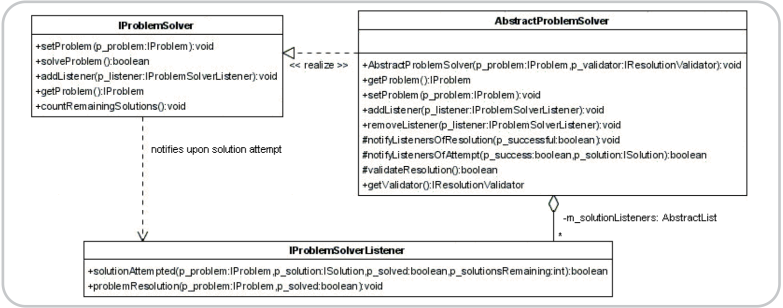

IProblemSolver
+setProblem(p_problem:IProblem):void
+solveProblem():boolean
+addListener(p_listener:IProblemSolverListener):void
+getProblem():IProblem
+countRemainingSolutions():void
<< realize >>
AbstractProblemSolver
+AbstractProblemSolver(p_problem:IProblem,p_validator:IResolutionValidator):void
+getProblem():IProblem
+setProblem(p_problem:IProblem):void
+addListener(p_listener:IProblemSolverListener):void
+removeListener(p_listener:IProblemSolverListener):void
#notifyListenersOfResolution(p_successful:boolean):void
#notifyListenersOfAttempt(p_success:boolean,p_solution:ISolution):boolean
#validateResolution():boolean
+getValidator():IResolutionValidator
notifies upon solution attempt
-m_solutionListeners: AbstractList
*
IProblemSolverListener
+solutionAttempted(p_problem:IProblem,p_solution:ISolution,p_solved:boolean,p_solutionsRemaining:int):boolean
+problemResolution(p_problem:IProblem,p_solved:boolean):void

Figure 5 The relationship between a problem solver and a problem-solver listener

```
public class StringBufferValidator
  implements IResolutionValidator
{
  public boolean validateResolution( IProblem p_problem )
  {
    NullStringBufferProblem issue =
      ( NullStringBufferProblem ) p_problem;
    if ( issue.isBufferNull() == true )
    {
      return false;
    }

    return true;
  }
}

...
// Prime the problem solver.
IProblemSolver solver =
  new AutomatedProblemSolver( NullStringBufferProblem,
                              new StringBufferValidator() );

...
// Validating inside an automated solver.
currentSolution.solve();
boolean resolved =
  m_validator.validateResolution( m_problem );
```

The system can provide listener objects to the problem solver to observe the resolution process, provide guidance to the problem solver, and adjust the environment as necessary in reaction to the outcome of any attempted solution. Often these components are specific to the operation at hand, but it would be reasonable to also make the observer interface implementations the same class that also implements the validating logic. Figure 5 illustrates the relationship between the problem solver and the problem-solver listeners.

IProblemSolverListener provides the final component in the resolution process that facilitates the application's active participation. Instead of determining whether the operation can be reactivated for further processing, concrete derivations of IProblemListener perform system housekeeping in reaction to the failure or success of the solution attempted. In many instances these listeners can record particulars of the error recovery attempt to facilitate the development of new system solutions that avoid the problem outright. Other listeners might attempt to reset the environment state if the process of executing a solution went awry. If the resolution attempt placed the task into a particularly dangerous or disrupted state, each listener has the opportunity to recommend to the problem solver that no further attempts be made (i.e., abort the task and let the system start over, even if that would result in lost data).

These suggestions are just that, however, and the ultimate responsibility for approving the abortion or continued processing of an operation remains with the problem solver's validator. For the best effect, a problem solver should notify each associated listener immediately after a solution is attempted.

```
...
// Inside an automated solver.
currentSolution.solve();
boolean resolved =
  m_validator.validateResolution( m_problem );
for ( int i = 0; i < m_listeners.count(); i++ )
{
  IProblemSolverListener listener =
    ( IProblemSolverListener ) m_listeners.get( i );
  listener.solutionAttempted( m_problem,
                              currentSolution,
                              resolved,
                              countRemainingSolutions() );
}
```

Conclusion

The tools necessary to identify the occurrence of errors are already available to developers, but additional work is necessary to provide a consistent and reliable resolution mechanism. A general error recovery framework can be easily created and extended to facilitate problem resolution in both Web and desktop systems. In many instances, however, expert insight into how the system works will be required to make the most of the framework's capability to ensure the best user experience possible.

This article explored the design of such a framework to provide a precise method for describing the problem at hand, identifying and attempting solutions, and ensuring that the problem has been resolved. The process can be automated in situations where user input is impossible or inappropriate, or made interactive through the presentation of dialogs and wizards. A complete implementation of this framework can be found at http://home.insight.rr.com/thebretts/todd/research/.

Reference

- Gamma, E., et al. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

This page is available in print only.

This page is available in print only.

JBuddy by Zion Software

Reviewed by
Ryan Cuprak

Just as the Web has revolutionized information distribution and retrieval, instant messaging is revolutionizing communication. Instant messaging is a powerful tool that few enterprises or application developers have fully harnessed. Despite its pervasive use by millions all over the world, few businesses have yet to exploit it.

Presently instant messaging in the enterprise is hobbled by competing service providers each with their own proprietary protocol. Most end users are wedded to one of the major providers, such as AOL, Yahoo, MSN, and ICQ, since networks of acquaintances create a formidable inertia to new players. If a business wants to provide services via IM, they have to play with at least the big three providers to reach customers. Complicating matters, existing IM clients are not extensible or easily integrated into business processes. Clients today do not centrally log messages or furnish the ability to expose enterprises or systems as one buddy and dynamically dispatch messages to potential responders based upon availability.

In the increasingly regulated post-Enron and HIPPA business environment where penalties are stiff, logging messages and ensuring confidential information isn't leaked is of utmost importance. This is where JBuddy comes to the rescue. JBuddy provides a uniform API against which developers can easily build their own applications that interoperate with instant messaging but serve their own needs.

JBuddy is an all-Java library that supports AOL Instant Messenger, ICQ, MSN, Yahoo Instant Messenger, and enterprise IM solutions including Lotus Sametime, Jabber, and Zion's own messaging protocol. For those developers required to use a non-Java language, JBuddy even sports a COM wrapper that exposes JBuddy in both COM and .NET using an interop layer. The details of the underlying protocols are cleanly hidden from the application developer and the API is deceivingly elegant and simple. JBuddy supports file transfer as well as the ability to set away messages, retrieve lists of buddies, check buddy availability, and receive system events from service providers. However, to tie into service-specific features, JBuddy provides an additional set of interfaces for each, allowing you to more fully leverage the different features. There are a total of 20 classes in JBuddy residing in two packages: com.zion.jbuddy and com.zion.jbuddy.filetransfer.

To put JBuddy through its paces, a small "BuildBuddy" bot was constructed that would send notifications when a software build was completed and respond to inquiries about the present build number. The notifications could be generated either by Ant tasks or shell scripts. At the start of a build the bot would change its status from "available" to "away" and broadcast a message to all buddies regarding either the success or failure of a build. A set of command-line programs to manage the bot's buddy lists and services were written to facilitate configuration. Since the bot should always be online, it was implemented as a server with clients connecting via RMI. Figure 1 shows the overall structure of the server. The BuildGateway class implements the JBuddy IGateway interface through which IM events are delivered. In the case of this bot, there's only one instance of BuildGateway created and it handles events for all registered services.

Zion Software, LLC

2842 Main Street
Suite 325
Glatstonbury, CT 06033
E-mail: info@zionsoftware.com
Phone: 860 432-6258
Web: www.zionsoftware.com

The source code for the bot can be downloaded from the ***JDJ*** Web site (www.sys-con.com/java/sourcec.cfm). To build and run the application you'll need Ant as well as a license from Zion for JBuddy. Compiling the project is easy, just type "Ant" within the project directory. To run the server you'll need to create a .java_policy file in your home directory with the following content:

```
grant {
  permission java.security.AllPermission;
};
```

In addition you'll need to edit the java.rmi.server.codebase property in the run_server.sh script to point to im.jar in the project directory. Once these modifications are complete, the server can be

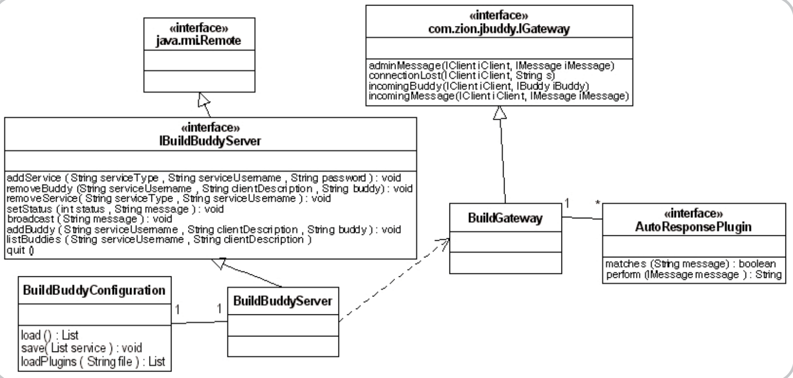

«interface»
java.rmi.Remote
«interface»
com.zion.jbuddy.IGateway
adminMessage(IClient iClient, IMessage iMessage)
connectionLost(IClient iClient, String s)
incomingBuddy(IClient iClient, IBuddy iBuddy)
incomingMessage(IClient iClient, IMessage iMessage)
«interface»
IBuildBuddyServer
addService (String serviceType , String serviceUsername , String password) : void
removeBuddy (String serviceUsername , String clientDescription , String buddy) : void
removeService(String serviceType , String serviceUsername) : void
setStatus (int status , String message) : void
broadcast (String message) : void
addBuddy (String serviceUsername , String clientDescription , String buddy) : void
listBuddies (String serviceUsername , String clientDescription)
quit ()
BuildGateway
1
*
«interface»
AutoResponsePlugin
matches (String message) : boolean
perform (IMessage message) : String
BuildBuddyConfiguration
load () : List
save(List service) : void
loadPlugins (String file) : List
1
1
BuildBuddyServer

Figure 1 Overall server structure

Ryan Cuprak is a software developer at TurboWorx Inc. He presently heads the Connecticut Java Users Group and has presented at both the CT JUG and New York City Java Users Group on Java Advanced Imaging (JAI). His interests include distributed computing, imaging, and Web application development. Ryan holds a BS in biology and computer science from Loyola University Chicago.

rcuprak@mac.com

started from the command line via "./run_server.sh", passing the hostname of the machine and optional plug-in configuration file. Table 1 lists the various command-line and Ant tasks. Before you can use either the command-line utilities or Ant tasks in a build process you must connect to a service by using "addService". AddService has the following parameters:

```
 addService <hostname> <client type> <user-
name> <password>
```

The hostname is the address of the machine on which the RMI server is running. The client type can be AIM, YIM, ICQ, MSN, etc., as found in the JBuddy Javadoc. Presently the bot will print out verbose output so you can see the callbacks in action. If the account you created doesn't already have buddies, you'll need to add them by using the "addBuddy" script. When adding a buddy you must specify the service type as well as the username of the account. Using the Ant tasks requires adding the task definition to your Ant file. The file "example.xml" is a prototype Ant script using the command-line utilities.

While the API is extremely concise, the devil is in the details. Since this is a fairly large application I'll focus on the core functionality as it relates to JBuddy. Although JBuddy does a great job of abstracting away the nastiness of the underlying communication protocols, managing the various clients with their idiosyncrasies is still a formidable challenge.

The most logical place to start is with the creation of a client connection. The "addService" method on the BuildBuddyServer connects to a service provider given the service type (AIM/YIM, etc.), username, and password. For ease of use, the settings for the service are saved as application preferences using the 1.4.1 Java Preferences API by the BuildBuddyConfiguration class. Connecting to a service is accomplished by passing an instance of IGateway into the factory method on IClientFactory:

```
IClient client =
IClientFactory.factory(<gateway>,<service
type>,<username><password>);
```

The service type is a constant from IClient such as AIM, YIM, ICQ, etc. The client created is not yet connected to the service. To connect to the service, the connect method needs to be called; however, it is an asynchronous method and when it returns you should check the return value first as you may not actually be connected to the service yet and hence cannot perform any operations on that client. Wait until the client is ready by checking the isOnline method before performing any operations on that client:

```
 client.connect();
 while ( !client.isOnline() ) {
Thread.currentThread().sleep(500);
}
```

Note: In a robust application the loop should also check for successful authentication to the service provider. If authentication fails, an admin message will be sent to the IGateway instance registered for the client with a text message of some sort that the username/ password is invalid, followed by a subsequent connection lost event to the gateway. The is Online method call would therefore block indefinitely. The message sent by the service is different for each service provider. The BuildBuddy bot doesn't handle this situation and assumes that the account and credentials provided are valid.

Inherited 1MegaSLOC of MegaSLOP?

Understand for Java

helps Java developers document, understand, and maintain impossibly large or complex amounts of source code.

Download and try on your code:
http://www.Scitools.com/

All downloads are fully functional, just time limited.

Scientific Toolworks, Inc.
sales@scitools.com
(435) 627-2529

Command-Line Tool	
addBuddy	broadcastMessage
removeBuddy	listBuddies
addService	quit
removeService	run_server.sh
Ant Tasks	
BroadcastTask	ChangeStatusTask

Table 1 Command-line and Ant tasks

During development, it's important to know that IM services, such as AOL AIM, impose several restrictions including message send rates, warning percentages, and limits on the number of times an IM account can log into the service within a specified time interval. If the threshold is exceeded, the account is inactivated for a period of time measured in minutes. This can cause significant problems during development and necessitates several backup accounts. All client objects are saved in a List to facilitate disconnecting from all of the services.

One of the core features of the bot was its ability to change "state" depending on whether a build was in progress. Since this bot maintains a presence on multiple services, it must iterate over the list, changing the state on each client object representing the connection. Interestingly, the capabilities of the various IM services vary dramatically with regard to away messages. Both AOL and Yahoo support custom away messages so that users can specify their own messages, whereas the other services have hard-coded choices or none at all. The status of a client can be set via:

```
client.setStatus(IBuddy.<type>,message);
```

The simplicity of this line is somewhat deceiving. The BuildBuddy bot has to set a different away type for each service. IBuddy.AWAY is used for AOL, whereas IBuddy.CUSTOM_AWAY is used for Yahoo and IBuddy.BUSY for MSN. In the case of MSN the away message is ignored. This is an area where perhaps another level of abstraction would be beneficial.

The broadcast method on BuildBuddyServer iterates over all the services and fires an instant message to each buddy. For each service there isn't a canonical list of buddies – it must be constructed by taking the distinct union of all buddy groups. A buddy can belong to more than one group; however, when blasting a message out to all buddies, sending the same message to the same person more than once is obviously not a good idea. Listing 1 constructs the unique buddy list for a given client.

After the unique list of buddies is acquired, a message can be sent out to each by simply calling "sendIM (<buddy>,<message>)" where <buddy> is a string. Each call to sendIM should be wrapped in a try/catch block since an IOException will be thrown if the buddy is not online. In the case of AIM, you can't repeatedly blast out instant messages to the same buddy in rapid succession. AIM has a time threshold for message intervals and if this threshold is violated, the client is automatically kicked off for a period of time. While this bot doesn't contain logic for spacing messages, any production system must take this into account or work out terms with AOL to have such restrictions lifted. In testing JBuddy I only ran into this when blasting the same buddy in a loop.

As mentioned previously it's through the IGateway callback interface that IM events are delivered. Whenever the status of a buddy changes, the "incomingBuddy" method will be called. The IBuddy object encapsulates the state information about the buddy. In addition, after the bot has sent a message to a buddy, if that buddy begins to type, a message of type IMessage.Typing is sent back to the bot. This is how regular clients know whether a buddy is typing a reply and thus is used to provide visual feedback. The adminMessage method receives notifications from the service regarding authentication and other service announcements. A call to the connectionLost method has a multitude of causes including the maintenance of a service's servers, loss of network connectivity, or the same IM credentials being used by another client. The BuildBuddy bot assumes that a lost connection stems from a server reboot, hence it attempts to reconnect. Because the connectionLost method is also used to inform the client of an authentication failure, this can be dangerous and additional state would need to be maintained.

One of the most exciting features of the bot that I will leave you to explore is the autoresponse plug-in architecture. You can add plug-ins that will respond to a query posted by a buddy. Each plug-in has an associated regular expression that determines to which messages it responds. Combined with the file transfer support APIs, there are unlimited opportunities.

There are some issues companies should be aware of when building applications on top of proprietary instant messaging protocols. First the various instant messaging providers are sensitive to security on their networks, having been bitten by viruses and "booter bots" (bots on the IM network that cause another user to be disconnected by exploiting a client vulnerability), thus they can and will unexpectedly modify their protocols. While there will undoubtedly be hiccups when protocols suddenly change, reverse engineering the changes is a much tougher challenge worth avoiding.

Zion has indicated that it was working with the different IM providers to forge agreements but no terms had been reached at the time this review was written. However, Zion will provide patches to subscription customers free of charge to correct incompatibilities as soon as they are available. In addition, JBuddy does not yet support retrieving/setting buddy icons. While this is purely a vanity feature, some may want the ability to choose an appropriate icon to represent their presence.

If you don't require public IM for your IM bot application or if you need to keep messages internal to your organization for compliance, etc., JBuddy SDK also supports enterprise IM solutions such as XMPP (Jabber), Lotus Sametime, as well as Zion's IM client as featured on java.com (JBuddy Messenger) and Java IM server (JBuddy Message Server), which will complete any realtime enterprise architecture.

Summary

JBuddy is a powerful library for building IM-enabled applications. Within minutes a developer can have IM capabilities built into an existing application. Enterprises looking to provide IM functionality for customers can use JBuddy as a solid foundation to interface existing services.

Listing 1

```
protected List getUniqueBuddies
(IClient service) {
List uniqueBuddies = new ArrayList();
     IBuddyList list =
service.getBuddyList();
      Enumeration grpEnum =
list.getListOfGroups();
     while ( grpEnum.hasMoreElements()
) {
        Enumeration buddies =
list.getBuddiesInGroup(((String)grpEnum
.nextElement()));
               while (
buddies.hasMoreElements() ) {
                      Object obj = bud-
dies.nextElement();
                       IBuddy buddy =
(IBuddy)obj;
                       if (
!uniqueBuddies.contains(buddy.getName()
)) {

uniqueBuddies.add(buddy.getName());
                       }
            }
     }
    log.info ( "unique buddies: " +
uniqueBuddies.size() );
      return ( uniqueBuddies );
      }
```

ODTUG NOW
2004
LET US SHOW YOU HOW
- To build a J2EE application with an Oracle database
- To integrate web services and Java components into Forms
- To utilize 9i/10g database features in application development
- To know when to use the various application development options
- To get the most out of Designer, Developer, and Warehouse Builder
ORACLE DEVELOPMENT TOOLS USER GROUP
Preconference Business Rules Symposium—Saturday, June 19
JUNE 20-24, 2004
THE WESTIN KIERLAND RESORT & SPA
SCOTTSDALE, ARIZONA
www.odtug.com

EXCLUSIVE Q&A WITH...

David A. Litwack

Senior Vice President of Web Application Development Products, *Novell*

Goodbye 'Middleware,' Hello SOA Applications

David Litwack is responsible for the development and advancement of Novell's secure Web services strategy, a position he assumed in July 2002 following Novell's acquisition of SilverStream Software, a company for which he'd served as president and CEO since 1997. He is also a member of Novell's Worldwide Management Committee. JDJ *speaks with him about a range of contemporary computing issues.*

***JDJ:* You've worked on PC products in the '80s, then pioneered client/server in the '90s. Since then we've had the Web, and now "Web services." Is it the right term, do you think? For example, an old Powersoft colleague of yours, Mitchell Kertzman, prefers to call the distributed application architecture "client/service." Do you agree – should we be talking about client/service architectures now?**

DL: Web services is a technology, not an application architecture. Client/server was comprised of a number of technologies, such as Windows, OO, SQL, ODBC, etc., which together allowed us to build applications in a new way. Web services – as the packaging, description, and discovery model – is only part of the new puzzle. It's the use of "services" as a component-based foundation for applications that is important. At its essence is a clean separation between the source of information and its delivery, which enables a far more flexible and personalized form of application.

There were two things wrong with client/server: the client referred to a specific hardware device and the server referred to a specific back-end system. There was a hard wiring between the two. The goal today is to provision any information or systems, regardless of how they are physically implemented, to any audience, regardless of how they connect, in a secure and personalized way based on identity. What we want is to dynamically match the logical service to an identity. In effect, identity/services is a better description of this new architecture.

***JDJ:* What about the buzzword of 2004, SOAs?**

DL: The industry has struggled a lot more with naming this architecture than client/server, maybe because it's a more comprehensive set of technologies. As I've just described, the essence is the service. So SOA is as descriptive as anything I've heard and seems to have broader acceptance today than any other term. I always try to refer to SOA rather than Web services. In fact, I believe you can be true SOA, without even being SOAP based.

***JDJ:* Still on Java – what's your position on the JCP – is it the right way to do things?**

DL: JCPs are time-consuming and complex, but so are all standards groups. I suspect the hidden agenda when the JCP question is asked is whether the JCP process is "fair," meaning, does Sun bias it? In fact it has been a fair and reasonable process, and Sun has been as reasonable a custodian as can be expected, considering the JCP is not a pure, open standard.

***JDJ:* Would/could anyone else safely be a custodian of Java? The open source community or IBM?!! How about Novell? ;-)**

DL: Tough question. Could Java move from Sun to a more truly open standard? Yes, I think so, with so many organizations already committed to the process. Could it move to open source? Maybe. But remember that you can be an open standard without being open source. Let's separate the two questions. I think Java becoming an open standard would be helpful with some of the industry politics and make Java generally more acceptable to everyone. It would also free up some Java-based efforts from some of Sun's restrictive licensing practices.

As far as open source, Novell has become a huge proponent. However, there are many flavors of open source, some more diffused in the community and some more focused in one or two organizations. Java and J2EE are huge and would require support from some large players. We could start with open sourcing some key pieces, like the JVM, if we could work around Sun's licenses.

***JDJ:* You have said that few Web services will be open source, since these are frequently tied to strategic, proprietary systems. Can we still expect that the presentation that consumes the Web service will be open sourced, though?**

DL: Remember that the essence of Web services is to black box a variety of back ends, to hide their technology. Inside these black boxes there is no standard. They could be mainframes, HP3000, relational databases, EDI, Web sites, SAP systems, and so forth. But on the consumption side, the world is becoming more ordered, with standards such as the Java Portlet spec and XForms. Novell has invested heavily in XForms, working with W3C, because we see it as the missing link that binds XML to presentation. A commonly accepted way of doing things is one of the elements that fosters open source.

***JDJ:* What's the overall effect on Java of the compelling economics of Linux?**

DL: Linux is clearly the next market wave. It's driven by the perception that open source has a better economic model for customers and frees them from vendor oppression. A lot of Linux will move into the corporate world in the next few years. Linux will not replace the mass of older systems out there. But there is an ethic about Linux that it will simplify and consolidate. Therefore, I believe that SOA will frequently ride into an organization on the back of Linux.

There are not as many new Java applications being written today as we would like because, frankly, there are not as many new systems being written, period. The adoption of Linux will drive an effort to simplify the historic IT mess, and SOA will be a big part of it. Java and J2EE are excellent environments for implementing SOA.

***JDJ:* How much closer are we to resolving the security aspects of Web services?**

DL: There are three things required to resolve Web services security: a general understanding of the issues; a universally accepted place where the solution will be determined; and acceptance and implementation by the industry. The first has largely occurred. The second involves a consolidation of sometimes competing standards groups. I think by the end of this year, the way will be clear for the delivery of all the key security standards, rapidly followed by commercial implementations by vendors like Novell.

***JDJ:* What about identity management, is it all sorted yet?**

DL: Identity management is yet another really broad term, with many facets. This category has grown up with big players starting from the directory/ metadirectory, and smaller players starting from a variety of areas, like password self-service, workflow-based provisioning, identity-based applications like white pages, and newer areas like virtual metadirectories.

Identity management mirrors SOA in some ways, with a number of moving parts that are fragmented but should really someday integrate to one thing. We're already seeing this consolidation occur, with a number of smaller players recently being acquired.

Ultimately, identity management should be about:

- Maintaining and administering a single view of identity, across disparate identity stores. This may include multiple directories or application data stores within or across organizational boundaries.
- Authorizing access based on role or organizational group, either via an administrator, delegate, or user self-service
- Provisioning resources based on centrally maintained policies or via workflow-based approvals
- Providing identity-based applications, such as white pages, yellow pages, org charts, skills inventories, etc., in an easily customized way that may also be embedded within applications
- Auditing and monitoring of all identity-based activities in a way that can satisfy regulatory requirements

It will increasingly become the industry view that this is one integrated whole, and not piece parts.

***JDJ:* Talking of complexity, is J2EE too complex? If so, what's the best way forward?**

DL: I believe that all standards go through three phases. First, the standard demonstrates its value but is immature, missing some of the basics, and we eagerly await the next version. Then the standard matures substantially, with many of the most frequently used pieces becoming robust. The standard is now enterprise ready, but it becomes harder for vendors to implement the much more complex standard, especially now that they have a large customer base to support and migrate. Finally, the standards body spends much of its effort on peripheral issues that a very small percent of the base will use or even understand. This occurs at precisely the time when the standard becomes mainstream and a core set of features are used widely, by mainstream users who hardly know about the more exotic features. At this point, vendors begin to question the need to implement the entire spec.

J2EE is well into the second phase. As the complexity increases, the relevance of incremental features diminishes and the standard starts to stabilize. An interesting side note with J2EE is that in the next year or so, you will see enterprise class, compliant, open source J2EE servers. It's possible, despite Sun's certification practices, that the pressure from the open source community for mainstream enhancements will trump the more theoretical nature of the mature standards committees.

***JDJ:* Why did Sun's Jonathan Schwartz say "Middleware is history"? Is middleware in fact just beginning? Or is Schwartz right, and end-to-end "systems" will supplant it?**

DL: What's in a name? SOA is an inherently middle-tier centric architecture. There's no doubt that in the world of SOA we will have application, integration, and portal servers; content management systems; policy and workflow engines; directories and metadirectories; identity providers; proxies; etc. These are all technically middleware. But as a market category, middleware may very well go away. Why? Because all of these things listed are only a means to an end and, therefore, not what people want to buy. People don't buy carburetors to have carburetors. Carburetors are a means to an end. People buy cars.

What is the equivalent of the car? An SOA application. At Novell, we've been working to bring together all aspects of identity-based SOA into a suite for that reason. The more transparent we can make middleware, the easier it will be to deliver SOA applications. That will be the new category.

From Within the Java Community Process Program

From set top boxes to wireless messaging

Onno Kluyt

Welcome to the May 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. Before we start, let me quickly follow up on last month's column: the Groovy JSR was approved unanimously. Congratulations to Strachan, et al, and I wish them much success with the JSR.

Coming Soon to a Television in Your Neighborhood

With a new JSR on the ballot, the JCP enters a new world. Cox Communications with the support of CableLabs, Comcast Cable, Time Warner Cable, and others have submitted JSR 242: J2ME Digital Set Top Box Profile – On Ramp to OCAP. The profile plans to target the J2ME CLDC platform and include APIs for I/O, networking graphics, and so on. It will also include a subset of the existing Java TV API. The JSR states that its goal is to address the current market of set top boxes, which have relatively small capabilities with regards to graphics, available memory, and processing. Another goal is for the profile to be forward-compatible with the more powerful boxes based on OCAP. OCAP stands for OpenCable Applications Platform. This is a Java technology–based middleware platform for the next generation of cable set top boxes for the North American market.

JDO, Take 2

JSR 12 defined version 1 of the Java Data Objects specification. Since its completion in April 2002 the technology has received a great deal of attention and feedback. Developers' experience with JDO has been incorporated into the submission of JSR 243, Java Data Objects 2.0 – An Extension to the JDO Specification. In version 2, the spec lead and the expert group are planning to improve JDO's alignment with the J2EE platform, improve ease of development, standardize the support for relational databases, and broaden the scope by including the persistence architectures from more vendors.

The "Almost There" Category

JSRs 86 and 205 are on the Final Approval Ballot, the last hurdle before a JSR is declared final. JSR 86, led by IBM, develops a specification for Enterprise Media Beans. This provides a framework to enrich applications that make use of entity beans in J2EE containers with media types such as audio, video, and images. The JSR standardizes the integration of encoding, decoding, and transcoding mechanisms into the J2EE application model. By being built on top of the J2EE technology, Enterprise Media Beans can provide a common model for security, persistency, and referential integrity.

JSR 205 develops the second version of the Wireless Messaging API and is led by Siemens. The first version, JSR 120, supported broadcasted and short messages. JSR 205 adds support for multimedia messages. The technology integrates into the security model of J2ME MID-P version 2.

"At JavaOne there will be the 'Java Communities in Action' event where the JCP, JINI, JXTA, and java.net communities will come together for a relaxing evening of networking, exchanging experiences, and discussing the highlights from each community"

The JCP @ the JavaOne 2004 Conference

Also this year the JCP will be present at the developers conference in San Francisco. On Tuesday, June 29, there will be the "Java Communities in Action" event where the JCP, JINI, JXTA, and java.net communities will come together for a relaxing evening of networking, exchanging experiences, and discussing the highlights from each community. Most likely this will be in the Argent Hotel. The program office will be staffing a station in the Sun booth in the expo hall; it's a prime opportunity to find me and my team and tell us how well (or not so well) we're doing. There will be a BOF focused on the JCP and we will be organizing a press panel together with Executive Committee members. We'll have a room set aside in one of the hotels where expert groups can meet during the week.

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

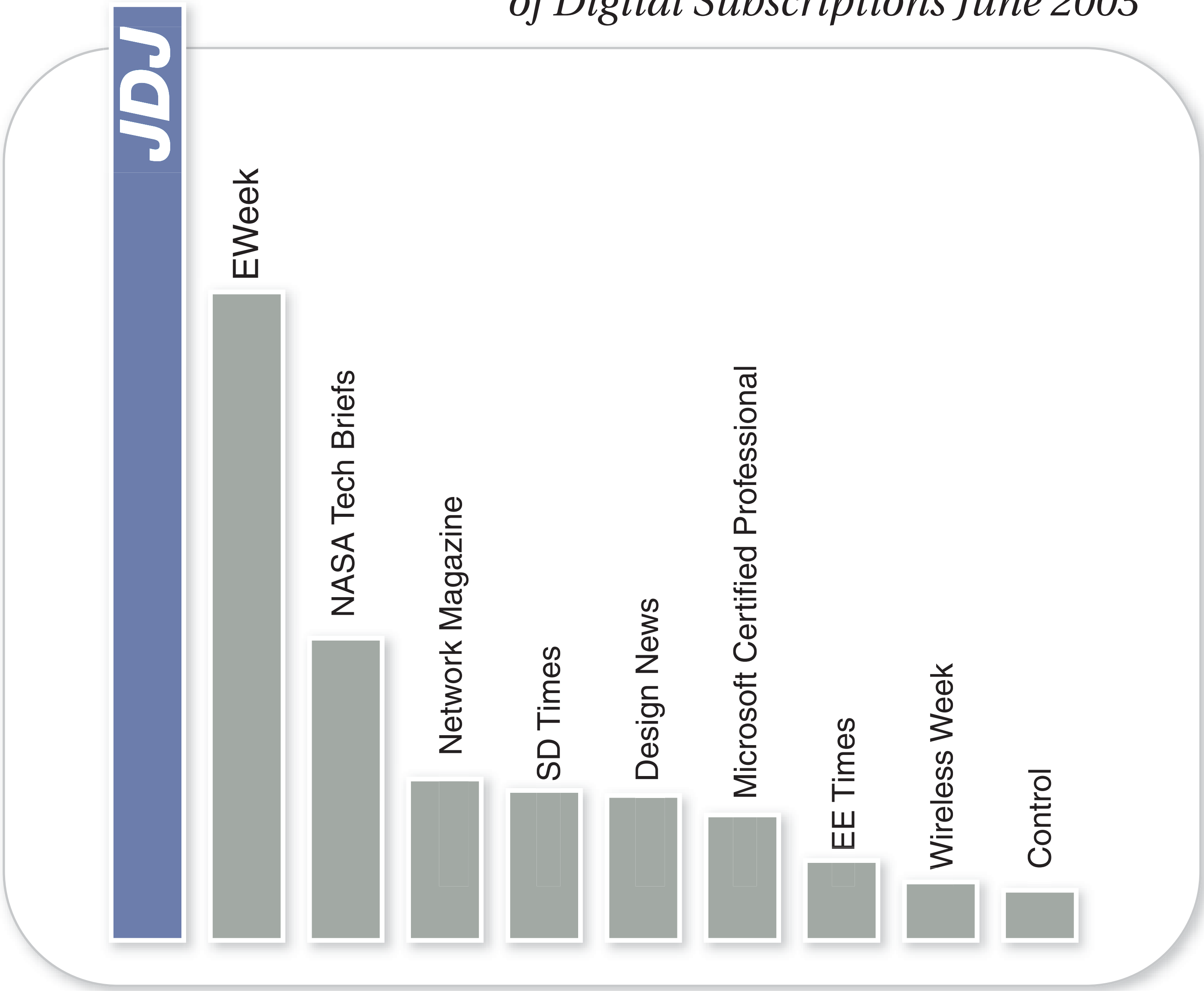
Onno Kluyt is the director of the JCP Program Management Office, Sun Microsystems.

onno@jcp.org

Oops, we did it again!

JDJ No. 1 in the World

Top 10 Publications Ranked by Volume of Digital Subscriptions June 2003



SYS-CON MEDIA

Source: BPA International publisher's statements
Note: Under BPA International rules, a subscriber who has opted to receive both the print and digital editions of a publication is reported only once and counted only once within that and the corresponding % of total qualified circulation of those combined subs are provided only for informational purposes.

Miles Silverman
VP Marketing

Carmen Gonzalez
Senior VP Marketing

Contact | Carmen Gonzalez: carmen@sys-con.com or 201 802.3021

JAVA DEVELOPER'S JOURNAL

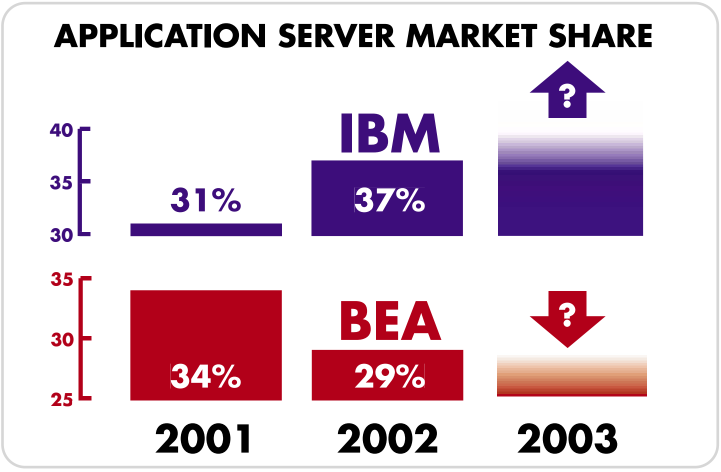
Gartner Report Due Out This Month:

How Long Can BEA Survive Against IBM?

The annual Gartner Research Report analyzing the application server market share among leading vendors is expected to be out this month.

According to the Gartner Report published last May, in 2002 IBM pulled ahead of BEA Systems for the first time and gained leadership of the J2EE application server market by walking away with a 37% market share; BEA dropped to second place with 29% of the market. The year before, in 2001, Gartner reported that BEA had 34% of the market share, with IBM close behind at 31%.

Recently, application server vendors have been extending their products to offer an application platform suite: a modern end-to-end platform for business applications. On the way, they are leaving behind the old standards and technologies.



BEA Systems has rewritten BEA WebLogic Integration to compete with leading integration specialists. Its key differentiating premise is the convergence of development and integration.

Having said that, when Gartner completes the tabulation of 2003 numbers and publishes their annual report, it may not be surprising to see IBM opening the gap further with WebSphere against BEA's WebLogic Application Server Platform in global market share.

This year's Gartner Report may show that roughly 25% of BEA's business depends on their partnership with HP, and roughly 12% on Sun-based installations.

Oracle has a very strong position with their database installations and should be taken seriously as a viable competitor, which should help them gain market share in the J2EE application server markets. Fujitsu Siemens Computers' (FSC) endorsement of Oracle Application Server 10*g* gives Oracle a strong partner for its application platform suite. Oracle will also benefit from FSC's mainframe integration technology. The latest version of Oracle's application platform suite (APS) has many new features beyond its much-advertised readiness for grid computing. "But enterprises should not rush any evaluation," according to Gartner.

JBoss uses its technical and business innovation in the J2EE application server market to take on the software industry giants. (see ***JDJ***'s April interview with David Skok).

Back to the two dominant players of the application server market: after currency calculations are worked into the 2003 market share estimates, IBM may further solidify its leadership position against BEA with a couple of additional points in gains. If BEA's 2003 numbers, published in their annual financials are normalized against the currency effect, their position could look even worse against IBM and may even look negative. Last year, BEA's $1 billion sales mark was greatly helped by the impact of the international currency translations of their financials. BEA Systems filed their annual report on April 15, 2004. IBM's market cap as of April 2004 was roughly $150 billion vs. BEA at around $5 billion.

Another caution that needs to be taken into consideration while analyzing market share is that this information is extracted from the companies' financial reports and determined as their actual application-server sales. There may not be a perfect way to know market share shifts other than what they provide.

Both IBM and BEA are doing a healthy amount of business in the Java space. In 2003 many solution provider partners bet on both companies by partnering with both until they saw signs of a clear leader arising.

In 2002, IBM also landed the No. 1 position in the portal space with 30.8% of the market, with BEA a distant second with 8% and Oracle third with 3.7% as well as dominating in message-oriented middleware with a whopping 80.8% of the market, according to Gartner. Sun's Sun ONE and TIBCO's middleware had 4.9% and 4.8% of market share.

The 2003 Gartner data is expected to be out by the time you read this article. Even if BEA closes the gap with IBM in the application server market share and gains back its leadership position, they will continue to be challenged by IBM, although in the short term this would eliminate any concerns to be flagged by Wall Street. On the other hand, if IBM continues to take market share from BEA, the survival odds may very well be against them in the long run, while competing with a Big Blue gorilla 30 times their size!

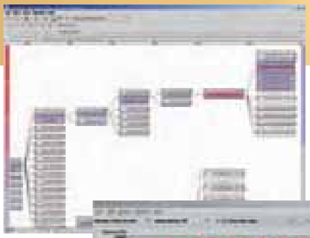
We will have an exclusive interview with Joanne Correia, vice president and research director at Gartner Group, analyzing this year's report, in our next issue. We will also interview others for their feedback on that.

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 trouble-shooting 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**.

QUEST SOFTWARE®