

HIT THE GROUND RUNNING WITH THE NEW ORACLE9iAS JAVA EDITION!

JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource
June 2003 Volume:8 Issue:6

www.JavaDevelopersJournal.com

Web Services Edge West 2003

web services **EDGE**
conference & expo

Sept. 30–Oct. 2, 2003
Santa Clara, CA

details on pg. 91



From the Editor

What's in a Specification?
Alan Williamson pg. 5

Viewpoint

Dial Tone for Business Apps
Glen Martin pg. 6

J2EE Insight

Too Much Innovation!
Joseph Ottinger pg. 8

J2SE Insight

Testing, Testing...
Jason Bell pg. 44

J2ME Insight

Interesting Technologies
Glen Cordrey pg. 62

JSR Watch

**From Within the Java
Community Process Program**
Onno Kluyt pg. 94

RETAILERS PLEASE DISPLAY
UNTIL AUGUST 31, 2003

\$5.99US \$6.99CAN



SYS-CON
MEDIA



by Joe Winchester
and Philip Milne
pg. 26



Tags: A Brief History of Tags
How to use a tag-based approach



Rich Rosen
10



Caching: Using JCache to Save Money
Application performance depends on efficient data distribution

Nigel Thomas
36



**XML & Java: XML Data Binding with
JAXB and UBL** Can you live without SAX or DOM?



E. Mooney & J. Fialli
46



Feature: Performance of Java Compilers:
An Empirical Study Microtuning your application

Haralambos Marmanis
52



Feature: J2ME Clients with Jini Services
Develop a highly portable, resilient service-oriented architecture



N. Patil & R. Dearing
64



Labs: P800 by Sony Ericsson
How well does the P800 perform when running Java apps?



Jason R. Briggs
72



Artificial Intelligence: Sex Machine Bean
A Java neural network built on the Agent Building and Learning Environment

Mike Fichtelman
76



Labs: Ensemble Glider by Ensemble Systems
An integrated development toolkit that accelerates J2EE development



Ron Phillips
82



Labs: ObjectAssembler 2.5 Enterprise Edition
by ObjectVenture, Inc. A visual "component" approach

Adam Chace
84

WE'VE ELIMINATED THE NEED
FOR MONOLITHIC BROKERS.
THE NEED FOR CENTRALIZED
PROCESS HUBS. THE NEED
FOR PROPRIETARY TOOL SETS.

Introducing the integration technology

YOU WANT.

Business Integration Suite



Introducing the Sonic *Business Integration Suite*. Built on the world's first enterprise service bus (ESB), a standards-based infrastructure that reliably and cost-effectively connects applications and orchestrates business processes across the extended enterprise. Extend your reach, cut costs and enhance your business agility. Thus eliminating one more thing - your headaches. To learn more, visit www.sonicsoftware.com.


sonic
SOFTWARE™

Connect_Everything.
Achieve_Anything.™



Why do industry leaders choose Zero G?

Because partnerships aren't a catch phrase at Zero G - they are our livelihood. We know that our success depends on yours. Whether you're a single-product developer or a multi-national corporation, we deliver the most innovative, scalable software deployment solutions in the industry - InstallAnywhere® and PowerUpdate®. But, more importantly, we deliver a team of skilled professionals who are committed to the partnership that starts the minute you download one of our products. That's why industry leaders like Sun Microsystems, Novell and Borland choose us.



Your software deployment partner
www.ZeroG.com

New JBuilder® 9



WORK THE WAY YOU WANT.
DEVELOP YOUR JAVA™ APPLICATIONS FASTER.
BUILD IN PERFORMANCE AND QUALITY FROM THE START.

Borland® JBuilder® makes the fast faster because it works the way you work. It knows what you need to do next, no matter what your style. JBuilder gives you an active voice at every stage in the development process and is the developer's access point into the Borland Application Lifecycle solution — best-in-class products integrated to help the whole development team make better software, faster. JBuilder — the #1 development environment for building any type of Java application: Web, Enterprise, Mobile, and Web Services.

Borland®
Excellence Endures™



go.borland.com/j1

DEFINE



DESIGN



DEVELOP



TEST



DEPLOY



MANAGE





Alan Williamson
Editor-in-Chief



What's in a Specification?

International Advisory Board

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

Editorial

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

Production

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

Writers in This Issue

- Jason Bell, Jason R. Briggs, Adam Chace, Glen Cordrey, Ron Dearing, Joe Fialli, Mike Fichtelman, Vlad Kolarov, Onno Kluyt, Babis Marmanis, Glen Martin, Philip Milne, Ed Mooney, Joseph Ottinger, Nikhil Patil, Ron Phillips, Rich Rosen, Nigel Thomas, Alan Williamson, Joe Winchester

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

Subscriptions

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

Editorial Offices

SYS-CON Media, 135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9600

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

©Copyright

Copyright © 2003 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission. For promotional reprints, contact reprint coordinator Carrie Gebert, carrieg@sys-con.com. SYS-CON Media and SYS-CON Publications, Inc., reserve the right to revise, republish and authorize its readers to use the articles submitted for publication.

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



This past month gave me a newfound respect for specification writers. I remember when James Davidson marshaled the early Servlet API and the lively discussions that ensued on the mailing lists, basically coordinating the entire operation (no JCP in those days, eh?!). The point is, at least there were discussions and, more important, a formalized document was produced at the end of the process. This past month I found myself dabbling in a number of technologies that sat outside the comfort of Javaland.

For a project I'm involved with (Blog-City), I was tasked with building the XML-RPC interface for the widely used Blogger API. This XML-RPC API has a handful of methods for updating your blog site remotely, enabling you to view recent posts, upload and edit posts, and all the normal things you would expect. All seems pretty simple so far, but the problem is that the documentation for this API is thin on the ground, and the one HTML page that does exist is over a year old. A trawl through the mailing list archives shows a lot of frustrated would-be users of their XML-RPC left out in the cold.

I did manage to figure it out eventually by looking up the source code from one of the SourceForge client projects to see what results they were expecting from the server. My aim was not to build a client, but the server side of the fence, so I wanted to make sure I caught all the different calls that could be sent to me. Although I have to say the irony wasn't lost on me that one of the world's leading blogging sites couldn't keep a page on their own API up-to-date!

Having now drafted the overall protocol flow, the job of actually providing the server functionality was next on my list. A quick look in Google and I discover the wonderment that is known as Apache's XML-RPC library. This was simply a joy to use. Take a single JAR file, easy-to-follow instructions, and within three lines of code, I

had a complete server running that accepted and processed XML-RPC requests. No special classes or interfaces to implement. No complicated descriptor files to maintain. It just worked, straight out of the box so to speak.

The whole process was a resounding success. But as the day ended, I reflected on the whole situation and how it highlighted the importance of standards bodies and specifications.

In many respects the Java world is lucky; by and large, we seem to do it right. We keep it open and try to involve as many people as possible. After my recent experience, I now think of Blogger as the Microsoft of the blogging world. There was no sign of any openness or even an effort to involve the community in the design and maintenance of their API. It was their way or the highway. Legions of client-side tools had come to accept the fact that they would need to keep an eye on the main server and continually update their own software for changes. Because one day, it just might not work.

As we know, such blind ignorance of the world around doesn't work. There are alternatives to the Blogger API, much better APIs in terms of documentation and proper version controls. These APIs, I am sure, will persist far longer than the Blogger API, simply due to the fact that there is a published specification.

For this reason, I am very happy to be in the Java camp as opposed to the C# (a.k.a. .NET) camp. At the end of the day, I feel secure knowing that when I call a particular method, it's going to be there the next time I make the call, and the next time, and so on. One day it isn't going to stop working because someone decided to tweak a method signature.

We take specifications for granted at times and, on behalf of the community, for all those who take the time to formalize the standards we rely on, I thank you. Keep up the good work and, more important, keep listening! ☺

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

alan@sys-con.com

Dial Tone for Business Apps



Glen Martin

Around 15 years ago there was a cascading switch failure in the telephone network along the Eastern Seaboard of the United States. Phones stopped working. So did air traffic control, because information about air traffic was communicated between control towers using the phone network.

The failure was caused by incompatible implementations of the inter-switch trunk protocols. These specs, while very long and detailed, are necessarily imperfect. With the best of intentions, different development teams interpreted the specs in different ways. When one switch started to fail, it sent out messages its neighbors didn't understand, and they failed. And so on, up and down the coast.

Such failures don't happen very often, because the standards are very good, the implementations are faithfully executed, and the systems are thoroughly tested. This stems from business motivations that make quality the highest priority.

When I moved from telecoms to databases, I was frankly appalled at the lack of formal specification and testing. Eventually I realized that this was an optimization with different goals. Business applications aren't like the phone network, which has to be all but 100% reliable – when was the last time you didn't get a dial tone? Failures are simply unacceptable.

Failures in e-mail and back-office applications are more acceptable in part because their scope is limited – between the applications are human correcting abilities and response times. When systems go down, we restart them and suppliers don't necessarily notice.

So why am I bringing this up?

Web services are turning back-office and desktop applications into something that's more like the phone network than what we're used to. Systems connected directly to one another lose those human correcting abilities and response times.

Reliability is more important in a world in which applications from different vendors written in different languages with different frameworks run-

ning on different hardware all have to interoperate correctly. Just as with the phone network, Web services will fail unless we have good specs faithfully implemented and thoroughly tested.

Good specs need to be thorough and unambiguous. If different "correct" implementations produce different results, how will programmers and consumers use them?

Faithful implementations need to follow the specs, but it is easy to find standards whose implementations are deliberately incompatible. HTML and Kerberos come immediately to mind.

The Java Community Process (JCP) addresses these issues by requiring that a specification be accompanied by a Technology Compatibility Kit (TCK) and a Reference Implementation (RI). The RI shows that the specification can be implemented, and to unearth ambiguities. The TCK verifies that the RI and any later implementations are faithful.

The value of Java standards depends on the completeness of these elements, and the integrity of those who implement them. If we measure that in terms of adoption, we're doing very well. Analysts estimate that there are about 3 million Java developers, and there have been over 1.3 million downloads of the J2EE platform 1.3 SDK.

There are around 15,000 tests in the Compatibility Test Suite (CTS) for J2EE version 1.3, and that number will increase substantially for version 1.4. The thoroughness provides assurance of portability of applications.

Application portability is the Java platform's "dial tone," thus the specifications are only available to those who commit to run the tests and ship only compatible implementations.

This is why developers and deployers need to be concerned with those who may claim to implement a specification, but who fail to demonstrate compatibility by running the tests.

Next time you're thinking about what platform to use for Web services, remember these questions: Is there an unambiguous specification? Is there a thorough test suite? Has your vendor run and passed the tests? ☉

Glen Martin is J2EE strategist at Sun Microsystems, and leads the marketing and product management team responsible for Java Web services and J2EE. Glen participated in the EJB expert group, and wrote the J2EE 1.3 requirements document and J2EE 1.4 concept document. He has 14 years of broad industry experience in technical and marketing roles, developing products ranging from packet switches to development tools and several points in between.

glen.martin@sun.com

President and CEO:

Fuat A. Kircaali fuat@sys-con.com

Vice President, Business Development:

Grisha Davida grisha@sys-con.com

Advertising

Senior Vice President, Sales and Marketing:

Carmen Gonzalez carmen@sys-con.com

Vice President, Sales and Marketing:

Miles Silverman miles@sys-con.com

Advertising Sales Director:

Robyn Forma robyn@sys-con.com

Director, Sales and Marketing:

Megan Ring megan@sys-con.com

Advertising Sales Manager:

Alisa Catalano alisa@sys-con.com

Associate Sales Managers:

Carrie Gebert carrieg@sys-con.com

Kristin Kuhnle kristen@sys-con.com

Editorial

Executive Editor:

Nancy Valentine nancy@sys-con.com

Associate Editors:

Jamie Matusow jamie@sys-con.com

Gail Schultz gail@sys-con.com

Jean Cassidy jean@sys-con.com

Jennifer Van Winckel jennifer@sys-con.com

Online Editor:

Lin Goetz lin@sys-con.com

Production

Production Consultant:

Jim Morgan jim@sys-con.com

Lead Designer:

Louis F. Cuffari louis@sys-con.com

Art Director:

Alex Botero alex@sys-con.com

Associate Art Director:

Richard Silverberg richards@sys-con.com

Assistant Art Director:

Tami Beatty tami@sys-con.com

Web Services

Vice President, Information Systems:

Robert Diamond robert@sys-con.com

Web Designers:

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

Accounting

Accounts Receivable:

Kerri Von Achen kerri@sys-con.com

Financial Analyst:

Joan LaRose joan@sys-con.com

Accounts Payable:

Betty White betty@sys-con.com

SYS-CON Events

President, SYS-CON Events:

Grisha Davida grisha@sys-con.com

Conference Manager:

Michael Lynch michael@sys-con.com

Customer Relations

Circulation Service Coordinators:

Niki Panagopoulos niki@sys-con.com

Shelia Dickerson shelia@sys-con.com

JDJ Store Manager:

Rachel McGouran rachel@sys-con.com



Joseph Ottinger

J2EE Editor

Too Much Innovation!

As I look over my choices for various tasks, I'm a little unsettled at how many choices I have, what they do, and how they interoperate. I'm not going to be the one to say that innovation is a bad thing, but too much innovation probably is a bad thing. In software design, it usually means the innovator hasn't looked into appropriate technology enough to know how to use what's available, so a new technology, a new mechanism, is invented. Witness BlueDragon, Vignette StoryServer, Velocity, Cocoon, XTP, CFMX, and JSP: all attempt to solve the same problem, albeit in different ways.

That means that people wanting to generate active content have a lot of choices: master CFML, Vignette's deployment, Velocity's templating syntax, Cocoon and XSL, Resin's XTP, or JSP's various oddities; we can throw in other variants like ATG Dynamo...it goes on and on, even without necessarily leaving Java. (Leave out the Java requirement and it gets worse: PHP, ASP, Tcl, mod_perl, CGI itself, etc.) No longer is generating content a simple decision, and while each technology has strengths and weaknesses, what I've found is that, in general, each "innovation" is a marketing tool, a result of laziness in researching available technology, or an attempt to lock in customers to proprietary mechanisms.

I've used most of these technologies, and I find myself using JSP for presentation, with WebWork providing the activation framework, and my own persistence framework handling data storage. Why? The real reason is because they're the best straight-line solution I see. I don't want to impress my peers with my continued mastery of technology for technology's sake; I want to impress my peers by not needing to trumpet how cool my tools are, by having a system that's under the radar. JSP may not be very impressive, but it gets the job done, and I don't have to spend time teaching people how to use it. WebWork takes a little getting used to for some people, but I've found that the payoff in discussing action invocations is well worth the time it takes. My persistence frame-

work (PortalWizard) is based on simple DAO abstractions. The innovation factor isn't very high, but then again, I'm able to roll up applications that are very flexible in a very short time.

I don't want to focus on presentation. I don't care, really, how my actions are called. Storage is something I only worry about if it's too slow or incorrect. I could try to innovate here; I could try to write a one-size-fits-all solution...but I don't care. I want to get the application working as a whole.

So when do I think you should innovate?

When you have little choice, that's when. The first step should always be investigation, and thorough investigation at that. That means actually using the technology at hand and pushing its limits to make sure it can't do what you need before you start blazing a new trail that won't help anyone in the long run. In general, what I've found is that people don't innovate to improve current technology; they innovate to see their names in lights, to impress others, or simply to get out of doing technology assessment. Innovation shouldn't be horizontal. It should be vertical. An innovation's strength should be so clear to those who understand it that using an alternate technology seems unsound. CGI was nice, for example, but having a way of executing content in the server is a *Better Thing*, because you don't waste time starting up new processes. This was an innovation, something new, something necessary. Coming up with lateral technologies that don't radically improve things is wasted time. (Consider Velocity, which I don't use - it's a templating mechanism that I should use, because it fills some needs I have in fantastic ways. It's lateral, but the way it renders is very nice. It gets my personal seal of approval.)

Innovation is good; it's even necessary. Nobody claims the technology we have is enough - not fast enough, not good enough, not complete enough. However, unbridled innovation is hurting our industry more than it's helping by breeding underpowered technology and clouding the market's vision. It's time to curb it. ☺

XML
Serialization of
Java Objects

26

Too Much Innovation!

As I look over my choices for various tasks, I'm a little unsettled at how many choices I have, what they do, and how they interoperate.



8

A Brief History of Tags

Now that Sun has provided some standards for custom tags in the form of JSTL, and has promised additional support for these standards in JSP 2.0, let's look at how we got to this point in tag history, and where we're going in the future.



10

Using JCache to Save Money

This article looks at some of the strengths and weaknesses of various caching architectures, examines how they fit into the surrounding J2EE and other ecosystems, and pinpoints each one's "sweet spot."



36

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

FIVE FACTS ABOUT WEB APP TESTING OTHER VENDORS DON'T WANT YOU TO KNOW.



With some Web app testing vendors, the thing to ask yourself isn't "what are they selling" but "what are they hiding"? Consider these five facts:

- FACT #1: It's hard to test dynamic, complex Web apps with a tool originally built to test client-server apps.
- FACT #2: It's even more complicated when you have to manually develop test scripts using a proprietary programming language.
- FACT #3: Developing separate scripts for functional tests, load tests, and performance monitoring is inefficient and unnecessary.
- FACT #4: You don't have to put up with restrictive licensing agreements, endless training, and expensive consultants.
- FACT #5: You can download our free evaluation and test it against your application in the same time it takes to watch their stale demo.

It's time to get the facts about Web app testing. See why IDC calls Empirix "a fast-growing leader in Web testing." * And learn how to get the same results as leading e-TEST™ suite customers like US Bank, British Telecom, and Ceridian.

Get your FREE Web App Testing Fact Pack, including:

- 10 Questions To Ask When Comparing Web App Testing Vendors.
- Newport Group Whitepaper: *Evaluating Critical Technology Differences Behind Automated Testing Tools*
- Empirix Whitepaper: *25+ Reasons Web Apps Don't Scale*
- Your own trial copy of e-TEST suite software

To get your Fact Pack:

- Call: 1.866.228.3781
- Visit: www.empirix.com/know
- Email: know@empirix.com



*Source: IDC Bulletin, The Distributed Automated Software Quality Tools Market Forecast and Analysis, 2001-2005, July 2001.

© 2003 Empirix Inc. Empirix and e-TEST suite are trademarks of Empirix Inc. All other names, products or services are trademarks or registered trademarks of their respective companies.



1.866.228.3781
www.empirix.com



A Brief History of Tags

Using a tag-based approach



Rich Rosen

Custom tags in JavaServer Pages have come a long way since their inception. Now that Sun has provided some standards for these tags in the form of JSTL (and the up-and-coming JavaServer Faces), and has promised additional support for these standards in JSP 2.0, let's look at how we got to this point in tag history, and where we're going in the future. (In addition, let's look at how we can use the JSTL taglibs and the Struts Taglibs that support the JSTL expression language right now.)

Tag-based approaches to Web application development are nothing new. Their origins can be traced back to HTML (since they mimic HTML's syntax), and are represented by such varied approaches as SSI, Macromedia's ColdFusion, Microsoft's Active Server Pages (ASP), and, of course, JSP.

JSP: Model 1 vs Model 2

One of the big problems with JSP Model 1 was that it lent itself to bloated "monolithic JSPs" that combine programming logic and presentation format in one module. Monolithic JSPs violate the principle of "Separation of Content from Presentation," to be sure. It's only when you have to maintain such JSPs in production applications that you begin to understand the importance of that principle in practice.

JSP Model 2 is an approach to Web application development that adheres to the Model-View-Controller (MVC) paradigm. Sun's vision for Model 2 is that the controller would be a servlet, the model would be represented by JavaBeans (or EJBs in more sophisticated applications), and the view would be comprised of JSPs that contain only presentation formatting constructs (i.e., no code). The presence of Java code in a JSP leads to the previously mentioned "monolithic JSP" syndrome, where data access and manipulation logic that belongs in the controller or model component of the application finds its way into the view component (the JSP). The intermixing of code with presenta-

tion formatting constructs results in a cluttered, unwieldy page that's not only difficult to maintain, it's not clear who is supposed to maintain it.

Custom JSP tags, a feature added to the JSP specification in version 1.1, makes it possible to achieve this desired separation of code from formatting. By encapsulating functionality in a single atomic entity that can perform complex processing that would have required a substantial amount of Java code, tags reduce (if not eliminate) the amount of code within a JSP page. (By code, I mean both scriptlets and the awkward JSP "expression" statements that make simple variable assignments more cumbersome than necessary.)

Well-designed tags allow a page designer to address and access data from the model that's constructed and manipulated by the controller. The decision about which data goes into the model (and which JSP view to employ for presentation) is in the hands of the controller. Thus, all a JSP developer needs to worry about is the layout of data already accessed and organized for presentation. (At least, that is the ideal – MVC frameworks encourage but do not enforce this separation.)

Historical Perspective: Programmatic and Template Approaches

Let's briefly examine how we got to where we are today, and what part tags played in this story.

Historically, Web application development approaches have fallen into two broad categories: programmatic approaches and template-oriented approaches. CGI (especially Perl CGI scripts) and the Servlet API fit into the programmatic category. Server Side Includes (SSI) and proprietary frameworks like ColdFusion fit into the template category.

The defining difference between these approaches is the focus of the page presentation object – the "view" in Model-View-Controller parlance. If the

primary content of the view component is code, interspersed with a few HTML formatting constructs, then we are talking about a programmatic approach. If the view component is basically an HTML page (or a page in some other target language such as WML or SMIL), with some embedded programming constructs, then we're talking about a template approach. More often than not, a template approach would make use of specialized tags that looked like HTML but were not part of the HTML language. These were server-side tags that performed conditional processing, iteration over a set of query results, and other complex application functions.

Naturally, these are arbitrary categorizations, and few approaches fit neatly into either one. PHP, for example, is often referred to as a template approach, yet a PHP module often contains more code than formatting. A ColdFusion module, on the other hand, looks structurally like an HTML page, but many also contain database queries in SQL.

Hybrid Approaches

Hybrid approaches abound, trying to be the best of both worlds by including page formatting and code constructs in one module. Microsoft's Active Server Pages and Sun's JavaServer Pages (specifically JSP Model 1) fit into this nebulous category, and both are, of course, very popular.

However, this combination of code and formatting in one module often hurts rather than helps. First, it's a violation of the aforementioned principle of "Separation of Content from Presentation." Having both content and presentation in the same module binds the content to a single presentation format. If a new presentation format is desired – either a new HTML page layout or a completely different format such as XML/XSLT – a new module that replicates a good portion of the old module must be created. Such replication naturally lends itself to problems



Developing Web Services before WebLogic Workshop.



Developing Web Services after WebLogic Workshop.

If getting dinner via a spear strikes you as less than efficient, you'll appreciate BEA WebLogic Workshop™. This unique development framework eliminates tedious and time-consuming steps, while you build Web services, create new applications and tackle the toughest integration challenges. Plus, you'll have all the support of our dev2dev developer community. To see how BEA WebLogic Workshop can radically simplify your Web services development, download it today at dev2dev.bea.com/useworkshop.



The fastest route to enterprise Web services.

as the application grows more complex. (The ability to include common page fragments that access and manipulate content helps us somewhat, but does not eliminate the problem.)

Then there is the question of “who owns this module?” Is it the Web designers, who are responsible for the page design and layout? Or the programmers, who are responsible for providing dynamic access to the data that populates the page? If designers want to modify the page layout, do they get to modify the ASP or JSP page themselves? If programmers need to change the code within an ASP or JSP page, what do they do if that change breaks the page presentation?

In practice, designers produce HTML mockups of pages that are then translated into JSPs by programmers. This means that even the slightest change to a page layout requires a remockup of the page layout that’s then retranslated into an ASP or JSP by programmers. What a waste of resources! Every time the page layout changes, a programmer must get involved, tweaking (or totally reworking) the ASP or JSP page. Why?

Because for all the hype about ASPs and JSPs being “friendlier” than pure programmatic solutions like CGI scripts and Java servlets, they aren’t so “friend-

ly” that you would hand them to page designers to work on themselves. This is partly because both ASPs and JSPs have historically required the inclusion of significant amounts of code to perform their tasks, and designers are generally not trained as programmers.

Tag, You’re It

It’s practically impossible to eliminate programming constructs from such a page entirely – how could we support variable content on a page (including a variable number of rows displayed from the results of a database query) without logic constructs that support iteration and conditional processing? Sure, we could write code that produced a complete HTML table containing the formatted results of a query, but this is another violation of the content/presentation separation principle. Programmers are writing HTML in their code – the very thing we’re trying to get away from. Furthermore, page designers have very little control (beyond CSS styles) over the presentation of an already formatted table.

Tag-based approaches are one solution to this problem. A small set of tags that support conditional processing, iteration, external resource inclusion, and common formatting tasks could be developed. Macromedia’s ColdFusion product, originally developed by the Allaire Corporation, gained popularity by providing such tags as part of its application platform. ASP and JSP followed suit, but did not provide all the necessary puzzle pieces (i.e., code was still needed within ASP and JSP pages to perform most nontrivial tasks).

Both Sun and Microsoft sought to fill that gap by offering developers the capability of designing their own custom tags. (ASP calls them “custom controls”.) This capability came without a lot of guidance from the vendors, and standards have been slow in coming. While a product like ColdFusion already had well-defined iterative and conditional constructs, JSP and ASP developers had to construct their own.

The developers of Struts, an MVC/Model 2 framework from the Apache Jakarta project, wrote their own set of tag libraries that provided a variety of powerful functionality. These included tags to perform processing logic (struts-logic), production of HTML elements (struts-html), and interaction with JavaBeans (struts-bean). They have been available since the release of Struts 1.0 and served to fill in many of the gaps that were pres-

ent in the JSP development process. As Struts’ popularity grew, so did the need for standard tags that performed these functions in a consistent way across all JSP-based applications, not just those written in Struts.

Sun finally came through with some guidance of their own: a specification for a standard set of tags known as the Java Standard Tag Library (JSTL). The tag specifications combine the functionality associated with some of the Struts taglibs with many of the original ColdFusion tags. In addition to the tags themselves, the JSTL specification also defines an expression language (EL) for accessing data components from the request context using a simplified notation that is a major improvement over code-oriented methods for accessing such data.

Jakarta developers who had been working on other taglib projects built a reference implementation for JSTL, which is now readily available and is included with many Jakarta project distributions. But JSTL is not the end of the story. Currently, there’s another project, called JavaServer Faces (JSF), that is seeking to standardize the presentation of page formatting elements in a fashion similar to the existing struts-html taglib.

Let’s focus on how the evolution of JSP custom tags has ameliorated the Web application development process by improving the structure of JSP pages to make them less code-centric and more MVC-compliant. JSTL is not a panacea, and JSF should not be expected to be one either, but looking at this evolution will help us understand where we are likely to go next.

Examples: Curing the Common Code

Let’s begin with a fragment from a classic Model 1 JSP page. While it’s deemed a good idea to eliminate Java code embedded within JSPs, this is easier said than done. Accessing JavaBean properties and including them in the page response, for example, can easily be accomplished within a scriptlet using JavaBean accessor methods and variable substitution.

```
<%
    MyBeanClass myBean = (MyBeanClass)
        session.getAttribute("myBean") ;
    String type = myBean.getType() ;
    String imageUrl = myBean.getImageUrl() ;
%>
<B>The value of the type property of myBean
    is <%= type %>.</B>
<P><IMG SRC="<%= imageUrl %>">
```

Installing JSTL and Struts Taglibs with EL Support

To use JSTL and the versions of Struts Taglibs that support the JSTL Expression Language:

1. Get the latest Struts distribution from the Jakarta project Web site (<http://jakarta.apache.org/struts>).
2. Copy all of the *.jar files out of the /lib directory into the WEB-INF/lib directory used in building/deploying your application.
3. Copy all the *.jar files out of the contrib/struts-el/lib directory into the WEB-INF/lib directory used in building/deploying your application. This will overwrite some of the JAR files you just placed there, replacing them with the EL-compatible versions.
4. Copy all of the *.tld files out of the contrib/struts-el/lib directory into the WEB-INF directory used in building/deploying your application. This will include TLD files for the Struts Taglibs, the versions of the Struts Taglibs that support the JSTL expression language, and the JSTL taglibs themselves.
5. When building your application’s web.xml file, provide Taglib definitions for the JSTL Taglibs and the versions of the Struts Taglibs that support JSTL EL, as shown in Listing 5.
6. Reference the appropriate TLDs in your taglib directives within your JSPs (see Listing 6).

Think you're using the best tool?



Think again.

intelliJ **IDEA 3.0**

Power-packed with unparalleled refactoring support, super intelligent code editing and completion assistance, a wide range of J2EE development features for rapid Web application and other enterprise development, a powerful Code Inspection tool, tight integration with Ant and JUnit, and a mountain of other productivity features for Java developers.

IntelliJ IDEA is simply the best Java development environment available.

IntelliJ IDEA, the integrated development environment for Java that will boost your productivity!

There are only 24 hours in a day. Use them wisely.

Download IDEA 3.0 and experience the only award-winning Java IDE that provides the ease-of-use, control and flexibility you demand, at a price you can afford.

Develop with pleasure!

Jet **BRAINS**

www.intellij.com

See us at JavaOne
Booth 1623

In the fragment above, a scriptlet accesses a session attribute (a `JavaBean` stored in the session with the name “myBean”) and establishes it within the page context. It then assigns the values of some of the bean’s properties to other variables. Using the `<%= ... %>` expression syntax, the values of these properties are included in the page.

Obviously this will not render reasonably when previewed in a Web browser. Not only are “type” and “imageUrl” variables whose values cannot be determined except in the context of a running application, but also their presence within the page is bounded by greater-than and less-than signs, just like HTML tags. Web browsers will see these and assume they have simply come across a new tag they haven’t been designed to handle. (The `<NOSCRIPT>` and `<NOEMBED>` tags make use of this feature to provide a degree of backward compatibility for older browsers.) Most browsers simply ignore such “tags” and pretend they aren’t there, thus an attempt to display this page in a browser will result in gibberish. Furthermore, the expression intended to display the image URL tag (“`<%= imageUrl %>`”) is embedded inside an HTML `` tag, which is a violation of XML formatting constraints.

Step 1: JSP Standard Actions

Sun created a set of JSP “standard actions,” which are tags that allow JSP pages to cooperate with `JavaBeans`,

```
is <jsp:getProperty name="myBean"
property="type" />.</B>
<P><IMG SRC="<jsp:getProperty name="myBean"
property="imageUrl" />">
```

This second example uses the XML directive syntax for standard JSP actions, including the `<jsp:useBean>` and `<jsp:getProperty>` tags. The `jsp:useBean` tag does what the first statement from the first example does: it accesses a session attribute named “myBean” and defines it to the page. Instead of using the awkward `<%= ... %>` syntax, this page makes use of `<jsp:useBean>` and `<jsp:getProperty>` tags.

This page has eliminated at least one problem: scriptlets have been removed from the page. Note, however, that the `` tag still contains a nested tag, `<jsp:getProperty>`, which is still a violation of XML formatting constraints. While most browsers today are tolerant of such violations in general, applications that generate XHTML for dynamic pages could not handle this notation. `<jsp:getProperty>` is in some ways an improvement over `<%= ... %>`, but it leaves many problems unresolved.

Although we have gotten rid of scriptlets for this example, it’s much more difficult to remove them from more sophisticated pages. Let’s imagine that “myBean” could have as one of its properties a `Java Collection` object that the page could iterate through. It may

sion embedded within it). Following this is a scriptlet block consisting of just the closing braces for the while loop and if statement. There is no “pretty” way to format “`<%>`” and “`%>`”, which are the strings that separate code from HTML in a JSP. Some code formatting standards exist that make the separation between code and formatting more distinct, but they are difficult to apply and don’t resolve the problem entirely.

After seeing code like that in Listing 1, you have to wonder about the assertion that JSPs are presentation-view components that could theoretically be constructed by Web designers. Looking at the attempt to include HTML formatting within this block (line 12), you might think we’d have been better off replacing that line with more code (i.e., an `out.println` statement), and forgoing the notion of embedding HTML entirely!

Step 2: Custom Tags (Struts Taglibs)

With the advent of JSP 1.1 came the ability to write custom tags. As mentioned earlier, Sun provided the capability, but not much in the way of guidance and standards. Along came the `Struts Taglibs`, which provided standard ways to perform iterative and conditional processing. They provided a reusable set of tags that could be employed in a variety of applications, and a simpler notation for accessing properties of referenced `JavaBeans`. Since the lack of tags to perform these functions was one of the main reasons that scriptlets were overused, the `Struts Taglibs` represented a big step toward reducing the amount of code within JSP pages (see Listing 2).

While this was a vast improvement over what came before it, even greater improvements were on the horizon. Sun got the message that providing the ability to write custom JSP tags was nice, but providing standard sets of tags to perform common functions was just as important. With this in mind, they wrote a specification for a `Java Standard Tag Library` (known as `JSTL`).

Step 3: JSTL

The `JSTL` tags fit into four categories: core (output, variable setting and removal, inclusion, iterative and conditional processing), XML (all of the above but with an XML orientation, plus `XSLT` transformation and `XPath` processing), SQL (creating database queries and processing result sets), and formatting (internationalization, local-

“ JSTL is not a panacea, and JSF should not be expected to be one either”

perform page redirection, and include external resources. In contrast to the original set of JSP directives (which used an ASP-like syntax beginning with “`<%>`”), these new tags adhere to an XML-compliant format. They use the “jsp” namespace to specify tags to define and access a `JavaBean` (e.g., `<jsp:usebean name="bean1" scope="session" ... />`), and access bean properties (e.g., `<jsp:getProperty name="bean1" property="prop1" />`).

```
<jsp:useBean name="myBean" scope="session"
class="mypackage.MyBean" />
<P>The value of the type property of myBean
```

or may not have this property, though, and one simple way to tell is the use of the `isMultiple()` method, which returns true if this `Collection` property is populated.

The page snippet in Listing 1 shows how that would be done using scriptlets. We have inserted the output associated with iterating over the collection between the section of the page displaying the “type” property and the section displaying the image.

To call this ugly is an understatement. There is a six-line block of Java code that precedes a single line of HTML (which itself has a JSP expres-

Application Server Experts Agree

“Enterprises that are seeking a well-appointed midtier solution will definitely want to give Oracle9iAS a test drive.”

 InfoWorld

“Oracle9iAS is now a formidable application server.”

 The MIDDLEWARE company

“Oracle: one stop app server shop.”

 eWEEK
THE ENTERPRISE NEWSWEEKLY

“[With Oracle9iAS,] Oracle is delivering what enterprises need to build a robust, scalable ebusiness infrastructure.”

 IDC
Analyze the Future

**Download a free trial
now at oracle.com/download.**

ORACLE®

Sources:
InfoWorld, November 2002
The Middleware Company, September 2002
eWeek, September 2002
IDC, May 2002

**oracle.com/experts
or call 1.800.633.1072**

Copyright ©2002, Oracle. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Reprinted from eWeek, September 2, 2002, with permission. Copyright © 2002 Ziff Davis Media Inc. All rights reserved.

ization, and other custom formatting). In addition, JSTL defines an expression language (often called “EL”) for accessing components directly in a class-agnostic, type-neutral manner, without excessive calls to intermediate tags and directives. For example, the notation “`#{sessionScope.brillig.borogoves[3].mimsy}`” addresses the property named “mimsy” from the third element in the indexed property called “borogoves” that’s found in the session attribute named “brillig”.

```
<@ taglib uri="/WEB-INF/c.tld" prefix="c" %>
<c:if test="#{sessionScope.myBean.multiple}">
  <c:forEach var="rnb"
items="#{sessionScope.myBean.collection}">
  <B><c:out value="#{rnb.name}" /></B>
  </c:forEach>
</c:if>
```

The page fragment above offers a small sample of the functionality available from JSTL.

The Jakarta Taglibs project has provided a number of other useful tags, some of which have historically overlapped with Struts Taglib functionality, but many of which are useful in their own right. They took it upon themselves to write the reference implementation for the JSTL tag set, which is available from the Apache project Web

Step 4: JavaServer Faces

JavaServer Faces (JSF) is a relatively new specification from Sun (JSR 127) that hopes to provide even more innovations. Though JSF is still an immature technology – an early access release was provided late in 2002 – it shows a good deal of promise. Pioneers who make use of this early release of JSF are virtually guaranteed to have their code obsoleted as the specification mutates and matures. Still, it’s a technology worth examining.

JSF appears at first glance to be a standardized plug-in replacement for the struts-html tag library, which provides a mechanism for pages containing HTML forms to interact with Struts Action classes. The goals for JSF go far beyond this. In fact, in the not too distant future, Struts will quite likely embrace both JSF and JSTL, converging on these standards instead of providing its own set of Struts-specific taglibs. JSF has more properly been called a cross between Struts and Swing, providing a framework for creating and manipulating generalized user interface components, including event handling and state management, in the context of an MVC-oriented Web application.

The capabilities provided with JSF include:

1. An MVC approach complete with a Struts-like controller servlet and request life cycle

only the beginning. Sun intends for vendors to provide custom RenderKits for other target formats, allowing the same JSF-enabled JSP page to be used for rendering in HTML, WML, SMIL, etc. (Note that to perform complex state management, event handling, and validation of UI components in HTML pages, JSF interacts with standardized JavaScript functions included on the page.)

Listing 4 is an example of a JSF page derived from the latest version of the JavaServer Faces Technology Tutorial available from Sun

(<http://java.sun.com/j2ee/javaserver-faces/docs/JSF.pdf>). The specification for JSF is changing rapidly, and it’s already known to be out of sync with the reference implementation, so use this example only as a guideline for what JSF pages are likely to look like when the technology matures.

Note that JSF tags are not only specialized in terms of the specific HTML element they correspond to (e.g., `<input type="text">`), but also in terms of what types of validation are performed for the tag. The `<h:input_number>` tag in the listing corresponds to an `<input type="text">` tag, but it’s also tied to JavaScript validation that ensures that the contents of the field are numeric. JSF includes not only form-oriented tags (`<h:input_*>`), but also tags for static display (`<h:output_*>`) and layout (`<h:panel_*>`).

While JSF technology is still in its infancy, it merits close observation as its specification becomes more stable. You can expect that increased synergy between JSF and JSTL will be a major factor in the acceptance of both technologies.

“Increased synergy between JSF and JSTL will be a major factor in the acceptance of both technologies”

site (www.apache.org/dist/jakarta/taglibs/standard-1.0/).

In addition, the developers of Struts also took it upon themselves to rewrite their own taglibs to use the JSTL expression language for parameter substitution. For backward compatibility, the original taglibs still exist and are the default set provided with the Struts distribution. To use the new versions of the taglibs that support the EL, special action must be taken when configuring your application, as described later. The page fragment in Listing 3 uses both JSTL core tags and the version of the struts-html taglib that provides EL support (the `<html:img>` tag) to provide a complete version of the original page fragment example using only JSTL and struts-html tags.

2. A standardized user interface component model that makes JSF roughly analogous to Swing for HTML pages
3. A validation framework providing functionality similar to Struts’ Validator API

JSF is not tied to a particular presentation technology (e.g., JSP), but it does provide a tag library of formatting-language-neutral user interface components for use in JSPs. In other words, while the tags in the struts-html tag library were bound specifically to HTML elements, JSF tags are agnostic about which target formatting language will be used to render your page. There is a default “RenderKit” that produces HTML 4.01, but this is certainly

Where Do We Go from Here?

1. Sun has already announced that JSP 2.0 will provide integrated support for the JSTL expression language anywhere within a JSP page. What would be nice is native support for JSTL, where the JSTL tags are as much a part of the core JSP syntax as `<jsp:xxx>` tags already are. This means developers and administrators would not need to take extra steps to install or configure these taglibs, or provide taglib directives for them within pages.
2. How about a tag that translates all EL expressions within the body? It seems like a real pain to write a `<c:out>` tag for every EL variable that’s being included in a presentation. You could include a whole block of text (containing multiple EL expressions) as the “value” param-

Rich Rosen works for the online edition of the *Wall Street Journal* as an application architect. He began his career at Bell Labs, where his work with relational databases and the Internet prepared him well for the world of Web application development. Rich is the co-author (along with Leon Shklar) of *Web Application Architecture: Principles, Protocols, and Practices*, which will be published by John Wiley & Sons later this year.

rlr@webappbuilders.com

ULC

Rich clients for J2EE

ULC offers rich client GUI components with a server-side programming model. Applications using ULC combine the benefits of browser and desktop applications.

ULC components are lean and based entirely on Java standards. They integrate with any existing J2EE software

Get this proven library and save time, money, and reduce your risk throughout the software lifecycle.

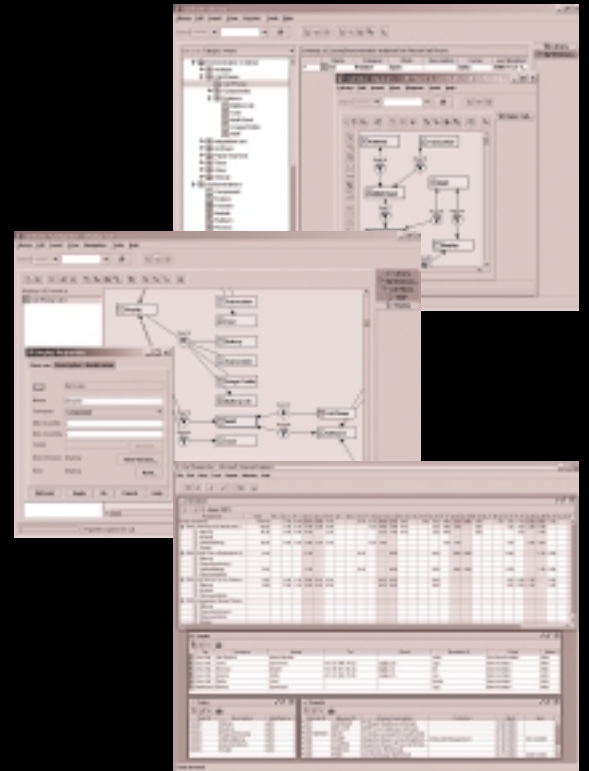
Benefits for:

Development & Maintenance

- rich GUIs with simple server-side programming model
- integrates with any J2EE software
- great tool support: visual editor, load tester

Deployment & Operation

- application release on server only
- application independent presentation engine on client
- minimal network traffic: several times less than HTML



ter of a `<c:out>` tag, but sometimes that's impractical (e.g., if the block of text contains HTML tags).

ColdFusion's `<CFOUTPUT>` tag turns variable substitution on for just the block of text found between its opening and closing tags (the "body" of the tag). A JSTL tag that does this same thing would be very nice indeed.

```
<c:output>
  Here is the image for
</c:output>
<B>${sessionScope.x.name}</B>:
  <IMG SRC="${sessionScope.x.imageUrl}">
</c:output>
  It can be found in the session attribute addressed by
  ${sessionScope.x}.
```

Yes, JSP 2.0 will provide this, but it will be a while before most JSP containers are compliant with 2.0 specifications. Furthermore, we might want to be selective about which parts of the page should participate in this variable substitution, e.g., the last line in the example above, where we want the string "`${sessionScope.x}`" to appear "as is."

- How about the ability to choose your own delimiters for the expression language? A notation like

``${pageScope.object.attribute}` may seem natural to Unix veterans, but (even though I qualify as one myself) I prefer a symmetric notation (e.g., `[[pageScope.object.attribute]]`), and I think many Web designers (who are supposedly the ultimate audience for JSTL-driven pages) are likely to agree. A possible syntax is offered in the following example.

```
<jsp:directive.taglibConfig el-delims="[[,]]" />
```

- While we're on the subject of Web designers and JSTL, how about integrated support for JSTL and the Expression Language in Web design tools? Dreamweaver and other popular tools already provide support for a variety of platforms including JSP, but a browser preview function that understands JSTL iterative and conditional tags and displays a reasonable mockup of the final page complete with "dummy" content would be a major step in enabling designers to gain full control over JSPs as presentation components.

Conclusion

In an MVC approach to Web application development, separation of content from presentation is critical. The

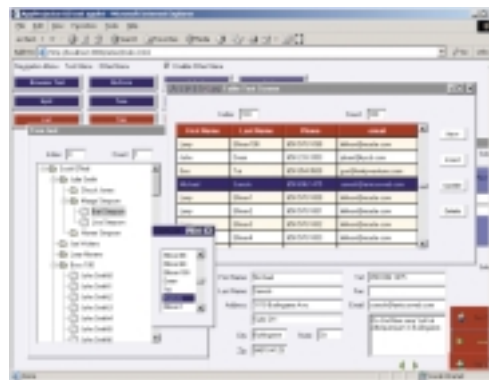
key to this separation is a clear definition of responsibilities, with programmers responsible for model and controller components, and page designers responsible for view components. Code embedded in view components, as found in monolithic Model 1 JSPs, makes it impossible for designers to have true autonomous responsibility for those components. A tag-based approach that eliminates code from JSPs facilitates the separation of responsibilities and enables each group, designers and programmers, to do their job without stepping on each other's toes.

We've only scratched the surface in describing the capabilities of JSTL. I've only demonstrated a small number of the available core tags and barely mentioned the XML, SQL, and formatting tag libraries. I have only given a brief nod to JavaServer Faces, the up-and-coming Sun-endorsed specification that standardizes and augments the functionality found in a number of the other Struts tag libraries. JSTL, JSE, and JSP 2.0 all represent major advancements in the arena of Web application development, but it is the synergy between them that will see the greatest advancements. ☺

Create enterprise web applications with powerful desktop-like user interfaces.



The Asperon AppProjector™ lets you easily create sophisticated web applications that have the full functionality of desktop applications and the ability to be deployed in a browser without a client install.



Get real-time interactive updates from server. Scroll through thousands of records in seconds. Powerful application model provides up to 10x reduction in development time.

ASPERON™
www.asperon.com

Download your free trial today!

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

JClass[®]

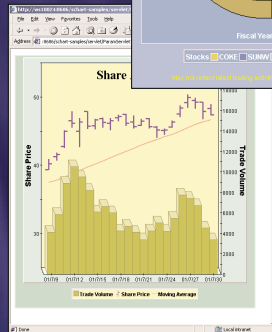
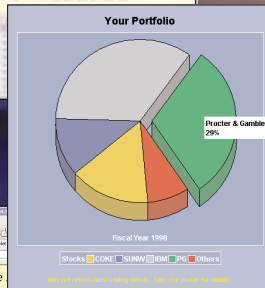
Rich client user interface and utility components.
Server-side web client interface and reporting
components. Whatever type of Java development
you're doing, JClass can help.

JClass ServerViews

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

JClass DesktopViews

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



See us at JavaOne!

Silver Sponsor, Booth #1501



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

Listing 1

```

1 <jsp:useBean name="myBean" scope="session" class="mypackage.MyBean"
  />
2 <B>The value of the type property of myBean
3 is <jsp:getProperty name="myBean" property="type" />.</B>
4 <
5 if (myBean.isMultiple()) {
6     Collection coll = myBean.getCollection() ;
7     session.setAttribute("collection", coll) ;
8     Iterator i = coll.iterator() ;
9     while (i.hasNext()) {
10         NestedBean nb = i.next() ;
11     }
12     <B><%= nb.getName() %></B>
13 <
14 }
15 }
16 <
17 <P><IMG SRC="<jsp:getProperty name="myBean" property="imageUrl"
  />">

```

Listing 2

```

<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
<bean:define id="myBean" scope="page" value="<%= session.getAttribute("myBean")
%>" />

<logic:equal parameter="myBean.multiple" value="true">
  <logic:iterate id="nb" items="myBean.collection">
    <B><bean:write name="nb" property="name" /></B>
  </logic:iterate>
</logic:equal>

```

Listing 3

```

1 <%@ taglib uri="/WEB-INF/c.tld" prefix="c" %>
2 <%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
3 <B>The value of the type property of myBean
4 is <c:out value="\${sessionScope.myBean.type}" />.</B>
5 <c:if test="\${sessionScope.myBean.multiple}">
6     <c:forEach var="nb" items="\${sessionScope.myBean.collection}">
7         <B><c:out value="\${nb.name}" /></B>
8     </c:forEach>
9 </c:if>
10 <P><html:img src="\${sessionScope.myBean.imageUrl}" />

```

Listing 4

```

1 <%@ taglib uri="http://java.sun.com/jsp/html/"
2   prefix="h" %>
3 <%@ taglib uri="http://java.sun.com/jsp/core"
4   prefix="f" %>
5 <jsp:useBean id="myBean"
6   class="mypackage.MyBean"
7   scope="session" />
8 <HTML>
9 <HEAD>
10 <TITLE>Sample JSP Page</TITLE>
11 <SCRIPT SRC="..."></SCRIPT>
12 </HEAD>
13 <BODY BGCOLOR="#f ffff">
14 <f:use_faces>
15 <h:form id="form1" formName="simpleForm">

```

```

16 <BR>Text Field: <h:input_text id="field1"
17   modelReference="myBean.attr1" />
18 <BR>Numeric: <h:input_number id="field2"
19   numberStyle="INTEGER"
20   modelReference="myBean.attr2" />
21 <BR>Password: <h:input_secret id="field3"
22   modelReference="myBean.attr3" />
23 <BR><h:command_button id="submit" label="submit"
24   commandName="submit" />
25 </h:form>
26 </f:use_faces>
27 </BODY>
28 </HTML>

```

Listing 5

```

1 <!-- struts-bean tags (with EL support) -->
2 <taglib>
3   <taglib-uri>/WEB-INF/struts-bean-el.tld</taglib-uri>
4   <taglib-location>/WEB-INF/struts-bean-el.tld</taglib-location>
5 </taglib>
6
7 <!-- struts-html tags (with EL support) -->
8 <taglib>
9   <taglib-uri>/WEB-INF/struts-html-el.tld</taglib-uri>
10  <taglib-location>/WEB-INF/struts-html-el.tld</taglib-location>
11 </taglib>
12
13 <!-- struts-logic tags (with EL support) -->
14 <taglib>
15   <taglib-uri>/WEB-INF/struts-logic-el.tld</taglib-uri>
16   <taglib-location>/WEB-INF/struts-logic-el.tld</taglib-location>
17 </taglib>
18
19 <!-- JSIL core tags with EL support -->
20 <taglib>
21   <taglib-uri>/WEB-INF/c.tld</taglib-uri>
22   <taglib-location>/WEB-INF/c.tld</taglib-location>
23 </taglib>
24
25 <!-- JSIL formatting tags with EL support -->
26 <taglib>
27   <taglib-uri>/WEB-INF/fmt.tld</taglib-uri>
28   <taglib-location>/WEB-INF/fmt.tld</taglib-location>
29 </taglib>
30
31 <!-- JSIL SQL tags with EL support -->
32 <taglib>
33   <taglib-uri>/WEB-INF/sql.tld</taglib-uri>
34   <taglib-location>/WEB-INF/sql.tld</taglib-location>
35 </taglib>
36
37 <!-- JSIL XML tags with EL support -->
38 <taglib>
39   <taglib-uri>/WEB-INF/x.tld</taglib-uri>
40   <taglib-location>/WEB-INF/x.tld</taglib-location>
41 </taglib>

```

Listing 6

```

<%@ taglib uri="/WEB-INF/struts-html-el.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/c.tld" prefix="c" %>
<%@ taglib uri="/WEB-INF/x.tld" prefix="x" %>

```



J2ME



J2SE



J2EE



HOME

PERFORMANCE IS EVERYTHING!

© 2002 Precise Software Solutions, Ltd. All Rights Reserved.

Give your J2EE applications lightspeed performance.

To excel in today's business environment, your J2EE applications need to scream like a bat out of hell. With heavy traffic bottlenecks and crawling end user response times, you're total roadkill. Proactive detection of application performance problems is crucial to rapid resolution, and to taking the detour around a crisis up ahead.

Precise/Indepth for J2EE correlates web, J2EE and database server performance to get to the root cause of performance degradation, from URL to SQL and beyond™.

As Precise/Indepth for J2EE anticipates performance degradation, it instantly notifies your top mechanics for analysis and tuning deep within the middle tier of your web infrastructure and provides automatic drill-down analysis and recommendations. This translates into streamlined J2EE web applications and lightning fast response times.

Turn up the speed limit for your J2EE applications. Talk to your Precise account representative today at 1.800.310.4777, or visit www.precise.com/jdj to download a free white paper on J2EE performance management and to register for an upcoming Webinar on J2EE application performance.



Performance is Our Business

Embed powerful reporting functionality
in your servlets, JSP, & Java applications
with the Formula One Reporting Engines!



DELIVER ESSENTIAL INFORMATION

Invoices. Statements. Activity Summaries.
Packing Slips. Transaction Histories. Reports.

EMBED AND DEPLOY FROM J2EE APPLICATION SERVERS

Servlets. Java Server Pages. Java Applications.

OUTPUT REPORTS IN ANY FORMAT USERS REQUIRE

PDF. Excel. XML. DHTML. HTML. Email.

CREATE NEW REPORTS QUICKLY, DYNAMICALLY

Visual Report Design Environments.
Programmatic Creation With APIs.

GET REPORTS UP AND RUNNING QUICKLY

Includes Sample JSP Portal. Deploy
on any J2EE Application Server.



Formula One e.Report Engine

For PDF, XML, HTML, DHTML and email reports.

Free Trials & Demos: www.ReportingEngines.com/info/JDJ_June_erc.jsp

Formula One e.Spreadsheet Engine

For richly formatted Excel reports.

Free Trials & Demos: www.ReportingEngines.com/info/JDJ_June_ese.jsp



888-884-8665 • 913-851-2200
sales@ReportingEngines.com





HOME



J2EE



J2SE



J2ME

We'll dispell two popular myths that have grown up around XML serialization: that it can only be used for JavaBeans and that all JavaBeans are GUI widgets”



XML

Serialization of

Java Objects

Long-term persistence of JavaBeans

by Joe Winchester and Philip Milne

Java serialization was initially used to support remote method invocation (RMI), allowing argument objects to be passed between two virtual machines.

RMI works best when the two VMs contain compatible versions of the class being transmitted, and can reliably transmit a binary representation of the object based on its internal state. When an object is serialized, it must also serialize the objects to which its fields refer – resulting in what is commonly called an object graph of connected components. Although the transient keyword can be used to control the extent to which the serialization process penetrates the object graph, this level of control is seldom enough.

Many have tried to use Java's serialization to achieve the so-called "long-term persistence" of data – where the serialized form of a Java data structure is written to a file for later use. One such area is the development tools domain, in which designs must be saved for later use. Because the logic that saves and restores serialized objects is based on the internal structure of the constituent classes, any changes to those classes between the time that the object was saved and when it was retrieved may cause the deserialization process to fail outright; for example, a field was added or removed, existing fields were renamed or reordered, or the class's superclass or package was altered. Such changes are to be expected during the development process, and any mechanism that relies on the internal structure of all classes being identical between versions to work has the odds stacked against it. Over the last few years the "versioning issues" associated with Java's serialization mechanism have indeed proved to be insurmountable

and have led to widespread abandonment of Java's serialization as a viable long-term persistence strategy in the development tools space.

To tackle Java serialization problems, a Java Specification Request (JSR 57) was created, titled "Long-Term Persistence for JavaBeans." JSR 57 is included in JRE 1.4 and is part of the "java.beans" package. This article describes the mechanism with which the JSR solved the problems of long-term persistence, and how you can take control of the way that the XMLEncoder generates archives to represent the data in your application.

We'll start our section by dispelling two popular myths that have grown up around XML serialization: that it can only be used for JavaBeans and that all JavaBeans are GUI widgets. In fact, the XMLEncoder can support any public Java class; these classes don't have to be JavaBeans and they certainly don't have to be GUI widgets. The only constraint that the encoder places on the classes it can archive is that there must be a means to create and configure each instance through public method calls. If the class implements the getter/setter paradigm of the JavaBeans specification, the encoder can achieve its goal automatically – even for a class it knows nothing about. On top of this default behavior, the XMLEncoder comes with a small but very powerful API that allows it to be "taught" how to save instances of any class – even if they don't use any of the JavaBeans design patterns. In fact, most of the Swing classes deviate from the JavaBeans specification in some way and yet the XMLEncoder han-



dles them via a set of rules with which it comes preconfigured. The XMLEncoder is currently spec'ed to provide automatic support for all subclasses of Component in the SDK and all of their property types (recursively). This means that as well as being able to serialize all of AWT and Swing GUI widgets, the XMLEncoder can also serialize: primitive values (int, double, etc.), strings, dates, arrays, lists, hashtables (including all Collection classes), and many other classes that you might not think of as having anything to do with JavaBeans. The support for all these classes is not "hard-wired" into the XMLEncoder; instead it is provided to the Encoder through the API that it exposes for general use. The variety in the APIs among even the small subset of classes mentioned earlier should give some idea of the generality and scope of the persistence techniques we will cover in the next sections.

Background

When problems are encountered with an object stream, they're hard to correct because the format is binary. An XML document is human readable, and therefore easier for a user to examine and manipulate when problems arise. To serialize objects to an XML document, use the class `java.beans.XMLEncoder`; to read objects, use the class `java.beans.XMLDecoder`.

One reason object streams are brittle is that they rely on the internal shape of the class remaining unchanged between encoding and decoding. The XMLEncoder takes a completely different approach here: instead of storing a bit-wise representation of the field values that make up an object's state, the XMLEncoder stores the steps necessary to create the object through its public API. There are two key factors that make XML files written this way remarkably robust when compared with their serialized counterparts.

First, many changes to a class's internal implementation can be made while preserving backward compatibility in its public APIs. In public libraries, this is often a requirement of new releases – as breaking a committed public API would break all the third-party code that had used the library in its older form. As a result of this, many software vendors have internal policies that prevent its developers from knowingly "breaking" any of the public APIs in new releases. While exceptions inevitably arise, they are on a much, much smaller scale than the internal changes that are made to the private implementations of the classes within the library. In this way, the

XMLDecoder derives much of its resilience to version-

older version. The XMLEncoder, by contrast, doesn't store a list of private fields but a program that represents the object's state. Here's an XML file representing a window with the title "Test":

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1" class="java.beans.XMLDecoder">
  <object class="javax.swing.JFrame">
    <void property="title">
      <string>Test</string>
    </void>
    <void property="visible">
      <boolean>true</boolean>
    </void>
  </object>
</java>
```

XML archives, written by XMLEncoder, have exactly the same information as a Java program – they're just written using an XML encoding rather than a Java one. Here's what the above program would look like in Java:

```
JFrame f = new JFrame();
f.setTitle("Test");
f.setVisible(true);
```

When a backward compatibility issue arises in one of the classes in the archive, it may cause one of the earlier statements to fail. A new version of the class might, for example, choose not to define the "setTitle()" method. When this happens, the XMLDecoder detects that this method is now missing from the class and doesn't try to call it. Instead, it issues a warning, ignores the offending statement, and continues with the other statements in the file. The critical point is that not calling the "setTitle()" method does not violate the contract of the implementation (as deleting an instance variable would), and the resulting instance should be a valid and fully functional Java object. If the resulting Java object fails in any way, an ordinary Java program could be written against its API to demonstrate a genuine bug in its implementation.

The vendors of popular Java libraries tend to devote significant resources toward programs to manage demonstrable bugs of this kind and enlist the support of the development community to work toward their eradication – Sun's "BugParade" is a well-known example. As a result of these kinds of programs, bugs that can be demonstrated by simple

"XML archives, written by XMLEncoder, have exactly the same information as a Java program – they're just written using an XML encoding rather than a Java one"

ing by aligning its requirements with those of developers who program against APIs directly.

The second reason for the stability of the decoding process as implemented by the XMLDecoder is just as important. If you were to take an instance of any class, choose an arbitrary member variable, and set it to null – the behavior of that instance would be completely undefined in all subsequent operations – and a bug-free implementation would be entitled to fail catastrophically under these circumstances. This is exactly what happens when a field is added to a new version of a class and this causes people to cross their fingers when trying to deserialize an instance of a class that was written out with an

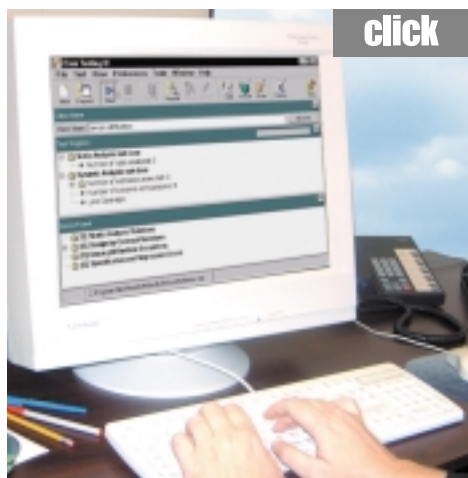
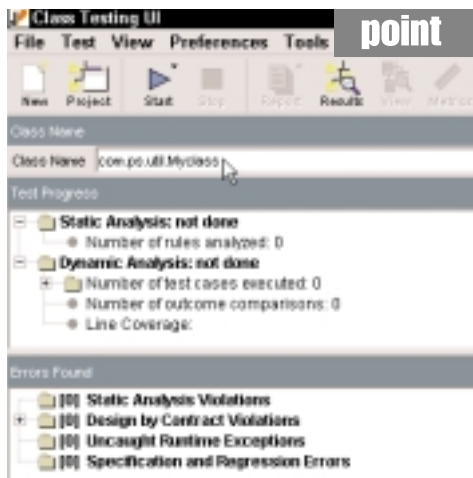
"setup code" tend to be rare in mature libraries. Once again, the XMLDecoder benefits here as it's able to ride on the coattails of the Java developer by using the public APIs of the classes instead of relying on special privileges to circumvent them.

Encoding of JavaBeans

To illustrate the XMLEncoder, this article shows serialization based on a number of scenarios using an example Person class. These range from simple JavaBeans encoding through nondefault construction and custom initialization.

In the simplest scenario, the class Person has String fields for firstName and lastName, together with get and set methods.

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



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

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

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

Learn how Jtest can enhance JUnit capabilities...

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

•----- **For Downloads go to www.parasoft.com/jdj2. Or call 888-305-0041.**

Special Offer:

For a limited time, every 100th customer to download an eval copy of Jtest automatically wins a fully licensed copy — FREE OF CHARGE.

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

Platforms:

Linux
Solaris
Windows 2000/XP

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



```
public class Person {
    private String firstName;
    private String lastName;
    public String getFirstName() { return firstName; }
    public String getLastName() { return lastName; }
    public void setFirstName(String str) { firstName = str; }
    public void setLastName(String str) { lastName = str; }
}
```

The following code creates an encoder and serializes a Person.

```
FileOutputStream os = new FileOutputStream("C:/out.xml");
XMLEncoder encoder = new XMLEncoder(os);
Person p = new Person();
p.setFirstName("John");
encoder.writeObject(p);
encoder.close();
```

The XML file created shows that Person class has been encoded, and that its firstName property is the string "John".

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1" class="java.beans.XMLDecoder">
  <object class="Person">
    <void property="firstName">
      <string>John</string>
    </void>
  </object>
</java>
```

When the file is decoded with the XMLDecoder, the Person class will be instantiated with its default constructor, and the firstName property set by calling the method setFirstName("John").

```
FileInputStream os = new FileInputStream("C:/out.xml");
XMLDecoder decoder = new XMLDecoder(os);
Person p = (Person)decoder.readObject();
decoder.close();
```

<http://java.sun.com/products/jfc/tsc/articles/persistence3/index.html>.

To serialize an object, XMLEncoder uses the Strategy pattern, and delegates the logic to an instance of java.beans.PersistenceDelegate. The persistence delegate is given the object being serialized and is responsible for determining which API methods can be used to re-create the same instance in the VM in which it will be decoded. The XMLEncoder then executes the API to create the prototype instance that it gives to the delegate, together with the original object being serialized, so the delegate can determine the API methods to re-create the nondefault state.

The method XMLEncoder.setPersistenceDelegate (Class objectClass, PersistenceDelegate delegate) is used to set a customized delegate for an object class. To illustrate this we'll change the original Person class so that it no longer conforms to the standard JavaBeans model, and show how persistence delegates can be used to teach the XMLEncoder to successfully serialize each instance.

Constructor Arguments

One of the patterns that can be taught to the XMLEncoder is how to create an instance where there is no zero-argument constructor. The following is an example of this in which a Person must be constructed with its firstName and lastName as arguments.

```
public Person(String aFirstName, String aLastName){
    firstName = aFirstName;
    lastName = aLastName;
}
```

In the absence of any customized delegate, the XMLEncoder uses the class java.beans.DefaultPersistenceDelegate. This expects the instance to conform to the JavaBeans component model with a zero-argument constructor and JavaBeans properties controlling its state. For the Person whose property values are supplied as construc-

“Although custom encoding rules can be supplied to the XMLEncoder, this is not true of the XMLDecoder”

To understand how to leverage the encoder and decoder for custom serialization requires an understanding of the JavaBeans component model. This describes a class's interface in terms of a set of properties, each of which can have a get and set method. To determine the set of operations required to re-create an object, the XMLEncoder creates a prototype instance using its default constructor and then compares the value of each property between this and the object being serialized. If any of the values don't match, the encoder adds it to the graph of objects to be serialized, and so on until it has a complete set of the objects and properties required to re-create the original object being serialized. When the encoder reaches objects that can't be broken down any further, such as Java's strings, ints, or doubles, it writes these values directly to the XML document as tag values. For a complete list of these primitive values and their associated tags, see

tor arguments, an instance of DefaultPersistenceDelegate can be created with the list of property names that represent the constructor arguments.

```
XMLEncoder e = new XMLEncoder(os);
Person p = new Person("John", "Smith");
e.setPersistenceDelegate(Person.class,
    new DefaultPersistenceDelegate(
        new String[] { "firstName", "lastName" }
    )
);
e.writeObject(person);
```

When the XMLEncoder creates the XML for the Person object, it uses the supplied instance of the DefaultPersistenceDelegate, queries the values of the firstName and lastName properties, and creates the following XML document.



JReport. Relied upon by millions of users everyday.

From the world's leading enterprises to your local school districts, millions of users rely upon JReport to bring critical information to the Web everyday.

JReport, the number one J2EE reporting solution lets you design reports effortlessly, requiring zero training. Your reports will go anywhere - whether through the Web, email or fax, requiring zero client software. Imagine pulling information from any data source, arranging it any way you wish, and delivering sophisticated reports precisely in HTML, PDF, Excel, XML, or any standard format.

Your Web users will enjoy highly interactive reports with all the drilldown, hyperlink, filtering and sorting capabilities. With the ad hoc reporting tool users can even modify or build new reports.

And all this is powered by JReport's 100% Java server engine, which delivers high performance, security, and easy integration with your enterprise systems. JReport's non-stop clustering technology gives you linear scalability as more CPUs and servers are added.

So what are you waiting for? Join the Java developers who are already taking advantage of JReport from Jinfonet - the leader in Interactive Reporting for the Web.



Interactive Reporting for the Web

See us at 2003 JavaOne, Booth# 707
Call us TODAY at 301-838-5560 or visit www.jinfonet.com/j6.htm

© Copyright 2003 Jinfonet Software, Inc. All rights reserved.
JReport and Jinfonet are trademarks of Jinfonet Software, Inc. Other trademarks belong to their respective holders.



```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1" class="java.beans.XMLDecoder">
  <object class="Person">
    <string>John</string>
    <string>Smith</string>
  </object>
</java>
```

The result is a record of the Object's state but written in such a way that the XMLDecoder can locate and call the public constructor of the Person object just as a Java program would. In the previous XML document where the Person was a standard JavaBeans component, the nondefault properties were specified with named `<void property="propertyName">` tags that contained the argument values.

Although custom encoding rules can be supplied to the XMLEncoder, this is not true of the XMLDecoder. The XML document represents the API steps to re-create the serialized objects in a target VM. One advantage of not having custom decoder rules is that only the environment that serializes the objects requires customization, whereas the target environment just requires the classes with unchanged APIs. This makes it ideal for the following scenario – serialization of an object graph within a development tool that has access to design-time customization, where the XML document will be read in a runtime environment that does not have access to the persistence delegates used during encoding.

Custom Instantiation

In addition to a class being constructed with property values as arguments, custom instantiation can include use of factory methods. An example of this would be if Person's constructor were package protected and instances of the Person class could only be created by calling a static `createPerson()` method defined in a `PersonFactory` class.

To write a persistence delegate requires a basic understanding of how the encoder creates its set of operations that will re-create the serialized objects when the stream is deserialized. The XMLEncoder uses the command pattern to record each of the required method calls as instances of the class `java.beans.Statement`. Each `Statement` represents an API call in which a method is sent to a target, together with any arguments. Commands that are responsible for the instantiation of objects are instances of `java.beans.Expression`. A subclass of `Statement` returns a value. Each object in the graph is represented by the `Expression` that creates it and a set of `Statements` that are used to initialize it.

For general control of instantiation, a subclass of the `PersistenceDelegate` class should be created with a specialized `instantiate()` method. The return value is the `java.beans.Expression` that indicates to the encoder which method or constructor should be used to create (or retrieve) the object. The returned `Expression` includes the object, the target (normally the class that defines the constructor), the method name (normally the fake name "new," which indicates a constructor call), and the argument values that the method or constructor takes.

The first argument of the `instantiate()` method is the instance of the Person object being serialized, and the second object is the encoder (see Listing 1).

When the XMLEncoder serializes the Person instance, instead of the `DefaultPersistenceDelegate` that uses standard JavaBeans rules for properties, it uses the anonymous inner class we registered as the persistence delegate of the `Person.class`. The resulting XML follows. In the `<object>` tag as

well as the class name, the static method `createPerson` has also been included, and the arguments are specified as child tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1" class="java.beans.XMLDecoder">
  <object class="PersonFactory" method="createPerson">
    <string>Smith</string>
    <void property="firstName">
      <string>John</string>
    </void>
  </object>
</java>
```

The inner class created for the Person persistence delegate subclasses from `DefaultPersistenceDelegate`, so the `firstName` property value of "John" is included in the XML document; however, no property tag is included for `lastName`. This is because the XMLEncoder compares the prototype instance of Person against the instance being serialized to determine which property values are not their default and need to be included in the XML document. The method that does this is protected `void initialize(Class type, Object oldInstance, Object newInstance, Encoder out)`. The `oldInstance` argument is the object being serialized and the `newInstance` is the prototype. Because the prototype instance is created using the `Expression` returned by the persistence delegate's method `protected Expression instantiate(Object oldInstance, Encoder encoder)`, the `newInstance` argument will already have the `lastName` set to be the same as the `oldInstance` so the encoder won't see their values as different and hence it does not serialize a property value for the `lastName`.

Custom State

The `DefaultPersistenceDelegate` assumes that the state of the `oldInstance` can be determined and restored by using the JavaBeans component model for properties. The list of properties for a class is retrieved using the method `java.beans.Introspector.getBeanInfo(Class aClass).getPropertyDescriptors()`. Each property is an instance of `java.beans.PropertyDescriptor` and includes a `get` and `set` method. The `Introspector` uses a set of rules matching method name pairs to create properties, although these rules can be overridden by supplying a specific `BeanInfo` class. The `BeanInfo` class can use a different set of methods than those that the `Introspector` would otherwise have determined as the property's `get` and `set` method. However, it can't deal with scenarios in which there is no `get` and `set` method, for example. For these the persistence delegate needs to be customized, and as an example we will have a property called `nicknames` that is multivalued.

```
private List nicknames = new ArrayList();
public void addNickname(String name){nicknames.add(name); }
public List getNicknames(){return nicknames; }
```

Nicknames are added to the class one at a time using the `addNickname()` method, and the complete list is retrieved using `getNicknames()`. The decoder needs to iterate through the nicknames and create an archive that uses the `addNickname()` method to re-create the Person.

The persistence delegate will subclass `DefaultPersistenceDelegate` that assumes construction of the class through a default Person, and will override the `instantiate()` method that's responsible for determining the expressions required to re-create the `oldInstance` (see Listing 2).

The persistence delegate iterates through the nicknames and for each one adds a statement to the encoder that speci-



Joe Winchester is a developer for IBM in Hursley UK, where he works on GUI software tooling for WebSphere. Joe was part of the expert group for JSR 57.



Philip Milne was the lead architect and expert group lead for JSR 57, and worked at Sun as part of the Swing development team. He now works as a consultant in London.



xmlspy⁵

WSDL Editor



Download
a FREE Trial
Now!

>>Modeling
of advanced Web
services is child's
play with the new
xmlspy 5 WSDL
Editor<<

Model Web Services

Web Service Description Language (WSDL) provides the power and flexibility to describe the interface through which a Web service is accessed. Yet WSDL is also complex, challenging even experienced software developers.

The xmlspy 5 WSDL Editor provides a sensible top-down approach to accelerate Web services development, allowing developers to model a Web service's interface in a language and platform independent manner, prior to writing any code. It supports visual modeling, editing and validation of WSDL files through an easy-to-use interface. Additionally use **xmlspy's** XML Schema Editor for defining WSDL types.

xmlspy 5 simplifies modeling of integrated enterprise systems, resulting in better software solutions at reduced costs and time-to-market. It's no wonder that **xmlspy 5** is the industry standard XML Development Environment used by a million software developers worldwide!

<http://jdj.altova.com/wSDL>

ALTOVA®

www.altova.com

fies the API to re-create the nickname. For this the Statement includes the target of the method (the Person oldInstance), the method name (addNickname), and the arguments (the nickname) (see Listing 3).

Specifying Delegates in BeanInfo Classes

In the examples used so far the custom persistence delegate was set directly onto the XMLEncoder by calling the method setPersistenceDelegate(Class,PersistenceDelegate). This works if you're the author of the code that's responsible for performing the serialization, but in some scenarios another piece of software such as an IDE tool is responsible for encoding the JavaBeans. In this situation you must teach the tool about the delegate that it should use for your class; this is done by specifying the delegate class name in the BeanDescriptor for a string key of "persistenceDelegate". For example, if the Person class is going to be introduced into an IDE together with PersonBeanInfo, the getBeanDescriptor() method would be specialized.

```
public class PersonBeanInfo extends SimpleBeanInfo {
    public BeanDescriptor getBeanDescriptor(){
        BeanDescriptor result = new BeanDescriptor(Person.class);
        result.setValue("persistenceDelegate", PersonPersistenceDelegate.class);
        return result;
    }
}
```

If the PersonBeanInfo is not in the same package as the Person class, the search path of the Introspector in the tool will need to be updated to include the BeanInfo's package.

Another way in which BeanInfo classes can be used to leverage persistence is by marking properties as transient. When DefaultPersistenceDelegate is responsible for encoding the JavaBean, it looks at all the available read/write prop-

erties and compares the existing values on the object being serialized against the values on the prototype instance. To flag a property so that it will be ignored, the key "transient" should be set to the value Boolean.TRUE. For example, if the "firstName" property should be considered transient, the getPropertyDescriptors() method on PersonBeanInfo could be specialized as shown in Listing 4.

Conclusion

This article explained how the design of the XMLEncoder avoids many of the fundamental pitfalls of binary serialization and makes the case that XML archives produced by the XMLEncoder can be trusted as a reliable means to store valuable data over the long term. Central to the design of the XMLEncoder is the java.beans.DefaultPersistenceDelegate class, which provides a default serialization strategy based on the idea of properties as laid out in a JavaBeans component model.

We show how custom delegates can be submitted to the encoder to teach it about idioms other than those of the JavaBeans component model, so classes that don't follow the JavaBeans conventions can be accommodated without changing their APIs. Because, in all cases, the decoder inflates object graphs using public API calls; deserialization is remarkably robust in the face of changes made to the classes referred to in the archives. If you need to save some critical data in your application to a file and are not interested in designing a new file format and coding the readers and writers for it – check out the XMLEncoder/XMLDecoder to see if they'll do it all for you. ☛

References

- *Using XML Encoder on the Swing Connection:* <http://java.sun.com/products/jfc/tsc/articles/persistence4/index.html>
- *JavaBeans:* <http://java.sun.com/products/javabeans/>

Listing 1

```
XMLEncoder e = new XMLEncoder(os);
Person p = PersonFactory.createPerson("Smith");
Cust.setFirstName("John");
e.setPersistenceDelegate(Person.class, new DefaultPersistenceDelegate(){
    protected Expression instantiate(Object oldInstance, Encoder out){
        String lastName = ((Person)oldInstance).getLastName();
        return new Expression(
            oldInstance,
            PersonFactory.class,
            "createPerson",
            new Object[] {lastName});
    }
});
e.writeObject(p);
```

Listing 2

```
XMLEncoder e = new XMLEncoder(os);
Person p = new Person();
p.addNickname("Jonny");
p.addNickname("Jonboy");
e.setPersistenceDelegate(Person.class, new DefaultPersistenceDelegate(){
    protected void initialize(Class type, Object oldInstance, Object
newInstance, Encoder out) {
        Person cst = (Person)oldInstance;
        Iterator iter = cst.getNicknames().iterator();
        while(iter.hasNext()){
            out.writeStatement(new Statement(
                oldInstance,
```

```
"addNickname",
    new Object[] { iter.next() }));
        }
    }
});
e.writeObject(p);
```

Listing 3

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1" class="java.beans.XMLDecoder">
  <object class="Person">
    <void method="addNickname">
      <string>Jonny</string>
    </void>
    <void method="addNickname">
      <string>Jonboy</string>
    </void>
  </object>
</java>
```

Listing 4

```
public PropertyDescriptor[] getPropertyDescriptors() {
    PropertyDescriptor[] result =
        new PropertyDescriptor[] {
            new PropertyDescriptor("firstName", Person.class),
            new PropertyDescriptor("lastName", Person.class)
        };
    result[0].setValue("transient", Boolean.TRUE);
    return result;
}
```


Rational XDE Professional Java Edition is:

UML modeling for Java

model ↔ code synchronization

automated design documentation

customizable design patterns

data modeling

and runs inside WebSphere Studio



In other words,
your IDE on steroids.

Lemme guess. You're thinking this is just one more tool that's actually going to get in your way. Hardly. Rational® XDE™ Professional blends right into your development experience. How? It runs inside IBM® WebSphere® Studio Application Developer, or stands alone with its own Eclipse-based Java™ Integrated Development Environment. It supports J2EE and J2SE. And Rational XDE support also includes membership to the Rational Developer Network,SM a helpful online community that provides reusable assets, Web-based training and discussion forums. We figure if someone else has already solved a problem, why should you have to do it again? To create a better development experience without creating new problems for yourself, get Rational XDE starting at only \$1,595. Or go to www.rational.com/offer/javacd2 to get a free Rational XDE Professional Evaluation CD so you can see it and try it out for yourself.



Using JCache to Save Money

What's the best solution for you?



Nigel Thomas

During the past 18 months, a rapidly growing number of organizations have been taking advantage of the emerging JCache standard for distributed caching to help scale application performance while at the same time reducing infrastructure costs.

This article looks at some of the strengths and weaknesses of various caching architectures, examines how they fit into the surrounding J2EE and other ecosystems, and pinpoints each one's "sweet spot." It will look at both "flat" and multi-tier frameworks, and contrast standards-based frameworks with proprietary offerings.

JCache: A Pluggable Java Temporary Caching Framework

The JCache specification (see sidebar) standardizes in-process caching of Java objects, and removes from the application programmer the burden of implementing standard cache features

More About JCache

JCache has been in development under the Java Community Process as JSR 107 (see <http://jcp.org/jsr/detail/107.jsp>) since it was first proposed by Oracle in early 2001. At the time of writing, the JCache expert group has not yet released a community draft, making it one of the slower JSRs in the pack.

There's always some risk in adopting a standard before it's ratified, but developers need not worry too much about the danger of being inextricably locked into a prerelease JCache API. First, the proposed API has been stable for quite a while now – although it has not been published outside of the expert group. Second, most developers will see only the client side of the API, which is not much different from working with regular Map objects – it's very easy to use, and it's also very easy to upgrade existing "roll your own" caches based on hashmaps.

At least one major user organization is represented on the JCache expert group alongside vendors such as Oracle, Gemstone, SpiritSoft, and Tangosol – and you can be sure that they wouldn't commit so much to the standard if they didn't think it was going to be worth it in the long run.

such as data validation, locking, eviction, and management (see Figure 1). As well as providing the basic put and get methods (a Cache extends a standard Map), the API offers a pluggable CacheLoader interface so users can add custom loaders for whatever data sources they are using.

An increasing number of products are supporting the JCache API; of course, simply supporting JCache is only a part of the solution. Depending on the problem you and your application developers are trying to solve, you should carefully consider which cache architecture best fits your requirements and constraints.

What Are the Trade-Offs Involved in Caching?

Caching is always based on compromise; a trade-off between performance, scalability, and accuracy using the various resources available. Ease of configuration is an important secondary consideration. Let's consider how we balance these factors to achieve the best performance possible.

Accuracy – How Stale Is Your Data?

When reading data from a cache for the second and subsequent times, how do you know whether the data is "stale"? Has the underlying data been changed? A cache can make one of the following assumptions:

- The data is valid "forever" (until the data item is ejected to make space for another, or the cache is closed down). This is easy to understand and implement but obviously only works for "static" data – today's weather forecast may be static, but a current stock price certainly isn't.
- The data is valid for a fixed period of time (or fixed number of accesses, or some other simple algorithm that the cache can apply), after which it is invalidated. This makes an excellent choice for data whose normal change cycle is longer than the "time to live" chosen, and where the inevitable (but occasional) use of

out-of-date data is not critical.

Neither of these approaches is perfect – either can leave a wide-open window of vulnerability during which stale data could be used by the application. We'll see later how active, pushed-based caches can reduce or remove that window.

Scalability – Will the Cache Itself Become a Bottleneck?

Although a cache is intended to improve performance, simplistic techniques can sometimes have a counter-intuitive effect. If you try to cache too much data, or if data is aged out too frequently, then the cache can add more overhead than it saves. Does your cache expand to such an extent that it is using up all your physical memory? If so, you may be spending far more time and effort paging (thrashing!) your virtual memory than you saved on data access.

Moreover, cache techniques that work well in single-user cases can break down when tens, hundreds, or thousands of users are involved. Multiple sessions serializing on synchronized accesses to the cache can be a significant drag on performance.

A single cache cannot grow indefinitely – sooner or later the workload somehow has to be spread across the network; thus the distributed cache is born. For read-only caches this is no big issue; each cache can operate independently, serving its own portion of users. Caches that need to support data updates meet additional problems of distributed locking and synchronization.

How Do Different Cache Architectures Measure Up?

Page Caches/Proxy Caches

Web server caches can appear within Web servers, or as stand-alone appliances in front of Web servers. Typically a Web server cache has a very simple model: supporting time-based invalidation, simple configuration policies

(based on file types, filename pattern matching, etc.), and perhaps offering some degree of operator control such as the ability to flush the cache (as a whole, or by region).

The main task of a Web server is to serve pages; the cache helps that along without requiring any complex programming. These caches work best when many users are accessing the same pages; all users get the benefit of the same cached pages.

Page fragment caches add a further refinement by allowing different rules

to be applied to different parts of the Web page. Static content is cached forever, volatile content has a time-to-live, and transactional content is served directly. Caching policy is associated with different page components using JSP tags, for example. These caches have to be more careful about data sharing, and typically they will have separate policies for application, session, and “global” data.

Database Caches

Most databases incorporate a data

cache of some sort; some include several. Oracle, for example, includes the “shared global area” that contains a cache of recently used database blocks as well as caches of compiled stored procedure code, parsed SQL statements, data dictionary information, and more. A correctly sized cache is a crucial component of a well-tuned database. However, you should realize that the process of extracting data from the cache is still very resource hungry:

1. The client application issues an SQL statement.
2. The statement is sent across the network.
3. The statement is compared to cached statements; if found in the cache there's no need to reparse it.
4. The parsed statement – with its generated access plan – is executed; the dictionary, index, and data blocks in the cache are searched.
 - Disk reads into the cache are made if necessary.
5. Row and column data is extracted from the disk blocks.
6. This data is finally sent back across the network to the client application.

Of these, only Steps 3 and 4 are affected by the cache. The other steps may well take several milliseconds and many thousands of CPU cycles. So there's still plenty of room for caching software outside the database, cutting out unnecessary calls to the data server.

Some products let you hide a further cache in the transport layers above the database; for example, there are a number of JDBC drivers that can cache the result sets from frequently executed SQL statements. These caches can cut out repeated reads, and are an easy retrofit to existing applications. However, they often don't deal well (or at all) with the problems caused when data is being updated as well as read.

Transactional Caches

This is where a transactional cache comes in – dealing with volatile data that's being created, updated, and deleted as well as read. Often the cache is linked to a particular programming model, either proprietary – often based on an object database – or based on standards such as J2EE's Container Managed Persistence (CMP), or the Java Data Objects (JDO) specification. The programming model provides the “ground rules” for the cache, identifying sessions, the start and end of transactions, and the locking policies to be used. Updates go through the cache to

Colorado Software Summit, the premiere international

Java & XML

programming conference.

www.SoftwareSummit.com

Quite simply, the Colorado Software Summit is the year's premier international JAVA & XML programming event. It's chock-full of real-world content for the serious Java and XML programmer — information you can actually use to move your career forward.

- Renown speakers & and a faculty of movers, shakers and problem-solvers
- A flexible, dream agenda of timely, bankable, make-you-a-better-programmer-today topics
- And the priceless chance to swap war stories with colleagues who “get it.”

To register or for more information call 800-481-3389 or 719-481-3389 for International calls.

Colorado Software Summit

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Kovsky Conference Productions Inc. is independent of Sun Microsystems, Inc.



edit-on® JavaBean SDK. Look no further!

More than 1000 companies, universities and organizations worldwide have chosen the RealObjects edit-on® Pro XHTML/XML content authoring applet as their in-browser WYSIWYG editor of choice.

Based on the same proven edit-on® technology, RealObjects is now offering the edit-on® JavaBean SDK. It enables professional Java™ developers to enhance their AWT- and Swing-based applications with WYSIWYG rich text editing capabilities within minutes.

What you
seek is
what you
get*

* RealObjects edit-on® JavaBean SDK enables WYSIWYG XHTML/XML content authoring. It brings word processor-like features to AWT- and Swing-based Java™ applications.

For more info and a free eval, visit:
www.realobjects.com



REALOBJECTS
Cross Platform Software

ensure that all parties (client sessions, the cache, and the underlying data store) are kept in sync.

Transactional caches can be pessimistic (lock early, which can result in serialization with sessions queuing for locks) or optimistic (lock as late as possible, which improves concurrency but is more likely to allow update conflicts to develop). Unless data is carefully partitioned, distributed locking adds latency; this increases the serialization effect. Anyone used to distributed databases will recognize the symptoms: a system reduced to a crawl while processors are still underutilized and sessions are simply queuing up on data locks.

Transactional caches tend to be tightly coupled to a particular application platform or coding style; some products require use of specific development and runtime tools, and it can be fearfully difficult to retrofit transactional caches onto existing applications.

Active, Push-Based Caching

Active caches turn the data concurrency problem around. Rather than having the cache try to predict whether cached data is still valid (the time-based approach) or check against the database (using transactional “select for update” locking), data updates can

be “pushed” out directly to the client caches, as well as “pulled” into the cache as a result of client application requests for data (see Figure 2).

The performance advantages are clear. Just one message is needed to notify a cache about new data values. Application threads register a listener with the cache, and the cache listens for the message. Client sessions are never at risk from stale, out-of-date data – whether the data changes once a day or every few seconds. Database access is reduced. Data is read once when cached; the “notification agent” – maybe a database trigger – fires only when the data is updated at the source. As long as updates are less frequent than queries, the push-based cache is extremely effective in reducing network traffic, and all data accesses except the first respond quickly, without network and datastore latency.

Heterogeneous Data Sources

Often there are several different types of data source; a homogeneous transaction model based on just one data type is inappropriate. Real-world applications deal with many different styles of data – relational, structured, object oriented – using many different access techniques – SQL, ISAM, LDAP, etc. It’s unusual for typical transactional caches to support more than one of these models cleanly.

An active cache based on JCache can sidestep the problem by providing a single API to cached data, across any data type, combined with the simple update notification interface. Data updates can be fed into the cache from the data server, or from its client applications.

Distributed Caches

When necessary, caches can easily be distributed across a server farm. The distributed caches can be:

- **Independent:** Each cache operates without reference to the others. The same data item may be cached in many places.
- **Partitioned:** Data is divided somehow between caches; clients (or the cache API) “know” which cache to address for each piece of data, which is held only once.
- **Coordinated:** The same data may appear in several caches, effectively side by side in a “flat” structure; cache misses may be served either from “peer” caches or from the underlying datastore. The cache software hides this complexity, and man-

ages the necessary exchange and locking of data.

Multi-Tier Caching – The Flexible Solution

It is also quite easy to develop “multi-tier” caches. At the bottom layer there’s a regular cache over the underlying datastore. Cache misses at this level convert to datastore lookups. To improve performance, further cache layers are added (see Figure 3); a cache miss at these higher layers converts to a cache lookup in the next layer down.

How does this help? Well, the top layer can be right up close to the application client – in the same virtual machine. This relatively small “VM cache” can be supported by a larger free-standing “local cache,” which soaks up most top-tier cache misses without needing any network traffic. The middle-tier local cache passes its own cache misses down to the bottom, much larger, datastore cache. With the added degrees of flexibility offered by two or three cache tiers, it’s quite easy to tune the caching framework to optimal performance for a specific application within the constraints of memory, processor, and network resources available.

Caching and JMS

Some distributed cache products use proprietary message formats, but increasingly JMS (Java Message Service) is recognized as being the best choice. Cache load requests are passed down the hierarchy using JMS queues, which can easily load balance requests; the data can be returned on a queue (for a specific cache) or on a topic (making it simple to organize cache clusters).

Update notifications can also be broadcast on a JMS topic; the notification agent is simply a JMS publisher client. Any kind of data source or data feed can easily be fitted into this model, and JMS guarantees to maintain the order of updates so that everything is kept consistent.

The alternatives to JMS are not attractive. Some use multicast, which is superficially attractive for broadcast – a single physical message reaches all intended subscribers – but does not offer guaranteed delivery, message ordering, or content-based addressing. Attempts to bolt these features on typically add more overhead than the multicast saves. Worse, deploying multicast across a WAN or the Internet is fraught with technical and administrative problems; many routers do not support (or do not allow) multicast traffic flow.

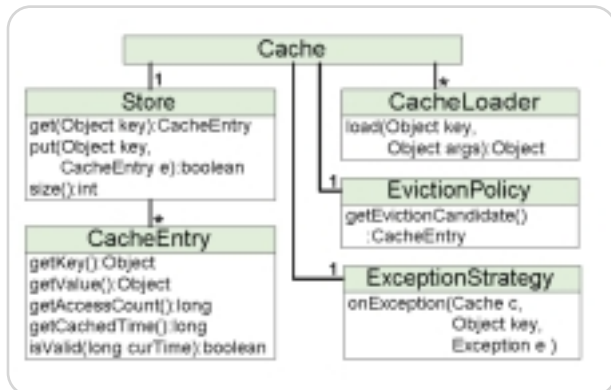


Figure 1 Cache and its pluggable strategies

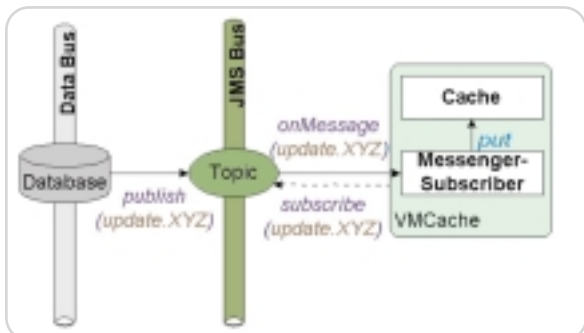


Figure 2 One tier – active cache update flow

Conclusion

The bottom line is that application performance depends on efficient data distribution. Since almost all server interactions involve data access, it's crucial to

Applications depending on JCache gain simplicity and flexibility in terms of configuration and performance management. By using a standard API, developers avoid the danger of being

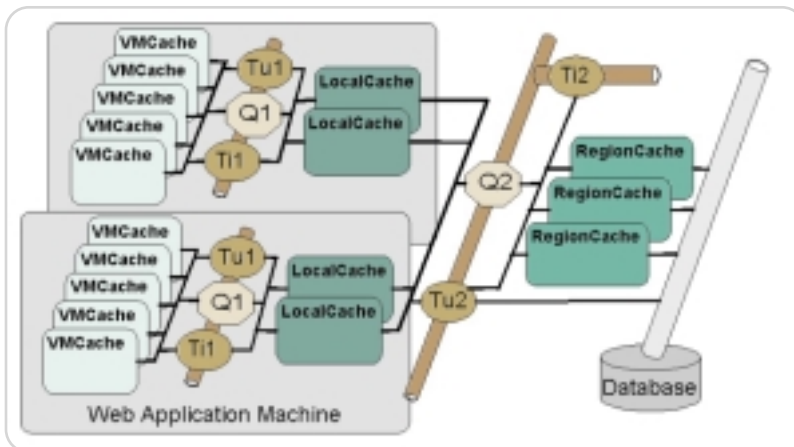


Figure 3 Three tier- the whole picture

ensure fast data access for maximum application performance. It pays to build a cache and avoid unnecessary round-trips to the datastore. By reducing traffic between the different layers of an application, you can substantially diminish the size and cost of the installation and greatly enhance the system's responsiveness.

locked into proprietary caching mechanisms. JCache can sit over any type of datastore – whether “static” (relational, object, or legacy databases, for example) or “dynamic” (for instance, a financial market data feed, process control telemetry readings, or network management events).

Adding JMS and an active push-based caching model into the equation lets the architect set the quality of service required, scale up to the load demanded, and fine-tune his or her intercache traffic. JMS traffic-shaping techniques – carefully honed to support enterprise messaging architectures – can be applied to optimize and balance network load in a multi-tier caching framework. Using JMS and JCache active push-based caching, developers can choose to cache anywhere in the application or network. In other words, caching can be done close to the data source or close to the delivery destination. Business requirements can then be more closely aligned with the enterprise architecture to ensure that caching is done at the optimal level. ☺

References

- Ross-Talbot, S., and Brown, G. “Scalable Web Services Using JMS & JCache.” *Web Services Journal*, Vol. 2, issue 3: www.spiritsoft.com/media/wsj/jms_jcache.pdf, www.sys-con.com/web-services/article.cfm?id=182
- eBizQ – “Standards-Based Caching Solutions for the Enterprise”: www.messageq.com/jms/spiritsoft_1.html

Nigel Thomas is
director of product
management at
SpiritSoft.

nigel.thomas@spiritsoft.com

Of course you can refactor Java code using just your IDE ...without RefactorIT™ ...



You can also remove your own appendix.

You already know you need refactoring to keep code flexible, maintainable, and performant. But most IDEs have sketchy refactoring features – if they have any at all. Only RefactorIT™ allows your team members to keep using the IDE each of them likes best, yet get **world-class refactoring features and metrics**. You'll improve development practices painlessly, and get common metrics across your entire team. RefactorIT™ plugs in to SunONE Studio, NetBeans, JBuilder, JDeveloper... or runs stand-alone with EMACS.

So get plugged into RefactorIT™, before you need to plug some other holes.



www.refactorit.com REFACTORIT™ IS A TRADEMARK OF AORIS SOFTWARE. ALL OTHER TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS.

You work with objects & xml. Why doesn't your report tool?

Object Reporting

ReportMill is the world's first XML/Object Reporting System. Today's web applications are built with objects and XML - now you can organize and display the data as well formatted web pages with one line of code.

Object Reporting means:

- No Redundant Database Connections
- No Redundant Data Fetching
- Reuse of Existing Business Logic
- Reporting from Any Data Source

Dynamic PDF & Flash Output

Real web applications demand professional looking output that draws customers to your website and inspires confidence in your product. HTML screen dumps are no longer the answer.

ReportMill generates true WYSIWYG pages in several different formats - all from the same template.

PDF for paginated documents that are convenient for viewing, archiving and printing
HTML/Flash for on-screen delivery of documents that are scrollable, interactive and portable



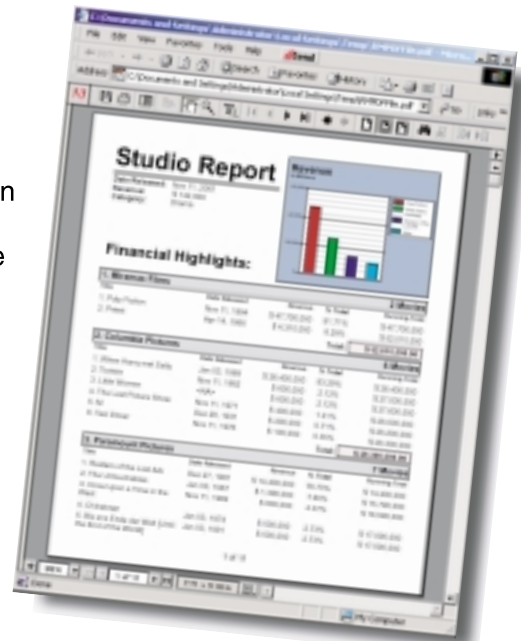
Drag n' Drop XML sample into new document and design template

Easy Integration

Creating web pages for your application couldn't be easier. ReportMill's page layout application is powerful and easy-to-use. The developer library has basically one line of API!



Fetch Objects and generate report in PDF and Flash



View in browser, email or write to file!

Supports ColdFusion, WebLogic, WebObjects, WebSphere and any Java based application.

Visit our website to discover how easy and powerful Object Reporting can be!

Fully functional evaluation version available at <http://www.reportmill.com/webstart>

For more information, call (214) 513-1636 or send email to info@reportmill.com.



ReportMill

ReportMill Software, Inc. Flower Mound, TX 214.513.1636 <http://www.reportmill.com>



Jason Bell
J2SE Editor

Testing, Testing . . .

While I was preparing for my interview with Bruce Eckel, a quote appeared in his Web log in May that said “If it’s not tested, it’s broken.” It got me thinking about how much I actually tested the code that I wrote. Now I don’t write JUnit tests for everything, but perhaps I should. To that end, here’s my proposal (I’m looking forward to the royalty check):

1. Sun should embrace the JUnit framework and include it in the Java Development Kit.
2. Any untested methods should throw an `UntestedMethodException`.

If that doesn’t tell you how much that download has been tested, I don’t know what will. Now I appreciate that there will be developer overheads in creating test cases, but perhaps it may make the job title “Test Engineer” popular again. This will decrease the number of unemployed IT staff; everyone’s happy – apart from developers.

It’s time to face facts, ladies and gentlemen; we got away with it for far too long. Now it’s time to change. Let me first reassure you that “change” is not a dirty word; in fact it’s one of the best things that can happen to you. Thinking differently forces you to apply yourself differently; take note of the implications of that change, and create a more secure, tight, and successful way of working. You feel good, the team feels good, and then management sees it and they feel good; company profits go up, the board and the shareholders feel good. All that from one small change. Only you control your destiny so make it work for you. Insist on change.

Anyway, back to testing. The time has come to put our former ways of testing behind us and look properly at what we do. Take the following example:

```
public class Calc {
    public Calc(){}
    public add(int a, int b){
        return a+b;
    }
}
```

```
public subtract(int a, int b){
    return a-b;
}
public static void main(String[] args){
    Calc c = new Calc();
    System.out.println(c.add(1,2));
    System.out.println(c.subtract(5,2));
    // oh! It works :)
}
}
```

It’s very easy to assume that this class has been tested and what I’ve been guilty of (I can only assume countless others as well; I just hope someone else confesses :-)) is accepting in my mind that it’s okay. What we really should be doing is supplying proof to everyone else that it works! I don’t want JUnit tests to be optional; I want them to be mandatory! Eclipse does a nice job of creating unit tests, but it would be nice if Eclipse (or insert the IDE of your choice) could monitor the methods and create unit tests for them automatically. Then all you’d have to do is create some test data and run the tests. Easy, simple, and you’d have evidence.

It’s all too easy to fall into the trap of delivering code that you think is okay, but in no time at all management is at your desk demanding to know why the application is not working. What evidence do you have with no unit tests? Effectively none. Now I openly admit that I have sat on the fence for far too long concerning this, and I think it’s one of the reasons that management spits fire and loathes some IT departments. At the end of the day there is only one person who can change my predicament – me.

Let’s put down our lattes/smoothies/stock options and start writing some proper unit tests, with decent test data and proper outcomes. I’m fed up with the stigma. If you want more info on JUnit, have a look at www.junit.org. There’s also an article in *JDJ* called “Test First, Code Later” by Thomas Hammell and Robert Nettleton (Vol. 7, issue 2) that’s excellent as well. ☺



Performance of
Java Compilers:
An Empirical
Study

52

Testing, Testing . . .

The time has come to put our former ways of testing behind us and look properly at what we do.

44

XML Data Binding with JAXB and UBL

As more and more industries standardize their data formats around XML, Java developers are challenged to keep up. XML data binding relieves the pain of any Java programmer who has ever winced at having to work with a document-centric processing model.

46

Jason Bell is a programmer and chief technical officer for a B2B Web portal in York, England. He has been involved in numerous Web projects over the past five years, the last two of which have been servlet-based.

jasonbell@sys-con.com

JSP | JDBC | CFML



JSP/Servlet Container

- Easy to install, configure, administer and deploy
- High-performance plug-in for Microsoft IIS and Netscape/iPlanet
- Flexible out-of-process configuration for IIS, NES/iWS and Apache
- Windows NT/2000/XP/2003, SPARC Solaris, Linux, and AIX

* Coming Soon: ServletExec 5.0

- JSP 2.0, Servlet 2.4
- JSTL 1.1, Java Server Faces (JSR 127)
- Web Services (JSR 101)
- Built-in Web Server



J2EE-Certified Type 4 JDBC Driver

- JDBC 3.0 (including RowSets)
- For connecting to SQL Server
- All Platforms (Pure Java)
- Hi Performance

BlueDragon™



CFML on J2EE

- Seamlessly migrate CFML (ColdFusion Markup Language) pages from proprietary ColdFusion servers onto any standard J2EE platform
- Deploys within standard J2EE WAR file
- Integrates with JSPs, Servlets, EJBs...
- WebLogic, WebSphere and any J2EE Server



In over 65 countries around the world, over 10,000 corporate and individual customers rely on New Atlanta products for their server-side deployments.

www.newatlanta.com

© 2003 New Atlanta Communications, LLC. All rights reserved.

XML Data Binding with JAXB and UBL

Process XML documents without SAX or DOM



Ed Mooney



Joe Fialli

As more and more industries standardize their data formats around XML, Java developers are challenged to keep up. That's especially true of developers employing a document-centric programming model, where an evolving schema can expose brittleness in your code and leave you wishing for a better solution.

XML data binding relieves the pain of any Java programmer who has ever winced at having to work with a document-centric processing model. Unlike SAX and DOM, which force you to think in terms of a document's structure, XML data binding lets you think in terms of the objects the structure represents. It does so by realizing that structure as a collection of Java classes and interfaces.

This is especially valuable when lots of applications use the same document schemas. Then the data binding approach yields a set of standard classes and interfaces that are reused across all the applications. This saves work since you don't have to write, debug, and maintain code to extract data from XML. There are even more savings if you're developing an application for one of the many industries that have agreed on standard XML Schemas for business data interchange: finance, travel, auto, and retail, to name just four.

This article will look at two new standards: JAXB and UBL.

SAX and DOM

SAX and DOM are among the oldest programming models for managing XML data (perhaps only younger than the "desperate Perl hacker"). SAX (Simple API for XML) is a stream processor for XML that requires you to implement the `org.xml.sax.ContentHandler` interface. This interface defines a set of callbacks that you register with the SAX parser. The parser calls your methods as it encounters various tokens in the input stream.

SAX is fast, uses memory sparingly, and is useful for accessing just part of a document. It's less efficient, though, if

you have to process the document more than once, or need to access document structures randomly, because it runs through the whole document instead of providing any navigational methods.

DOM (Document Object Model) provides a set of APIs for accessing an in-memory representation of the XML document. This representation forms a tree, which the application walks through looking for relevant information. DOM is useful if you need to edit an XML document, or access parts of it randomly. However, DOM implementations tend to be memory intensive, and sometimes even need to store the entire document in memory. This may not even be possible for very large documents.

Frequently, the goal of both SAX and DOM programmers is to initialize domain-specific objects with XML data. For example, given the element:

```
<Name first="John" last="Smith"/>
```

a SAX programmer would have to write a content handler and a DOM programmer would have to walk a parse tree to initialize an instance of this domain-specific class:

```
public class Name {
    public String first;
    public String last;

    public String getFullName() {
        return first + " " + last;
    }
}
```

Having this instance gives other parts of the application access to XML data using domain-specific logic. However, you get there only by first dealing with the document's structure by way of SAX or DOM. XML data binding, on the other hand, automates this step.

XML Data Binding

XML data binding binds XML constructs directly to objects. Several open

source projects for binding XML to Java have taken somewhat different approaches. The superset of their capabilities includes:

- Generating Java source code from an XML document
- Generating Java source code from an XML Schema (either DTD or XSD [XML Schema definition language])
- Generating an XML document from an arbitrary set of Java objects
- Unmarshaling an XML document (creating in-memory objects)
- Marshaling an XML document (writing out an XML document from in-memory objects)
- Validating an XML document, in its unmarshaled or marshaled form

A recent standard, JAXB, standardizes the XML data binding interface.

Java Architecture for XML Binding

Java Architecture for XML Binding (JAXB) was developed in Java Specification Request (JSR) 31. It was written by an industry expert group under the auspices of the Java Community Process. By standardizing the XML data binding interface and providing a conformance test, JAXB allows you to choose among different XML binding implementations without having to rewrite your application. JAXB also comes with a standard implementation, which we'll use to show you how to bind the UBL schema to Java objects.

UBL

Universal Business Language (UBL) is an XML-based business language built upon existing EDI and XML business-to-business vocabularies. It's the product of the UBL Technical Committee of OASIS. The committee intends to have UBL become an international standard for electronic commerce. If you're a J2EE programmer, there's a good chance UBL will be a part of your future.

The latest UBL 0.7 release (see References) contains schema, sample

Ed Mooney is a staff engineer at Sun Microsystems, Inc. He's currently the project lead for the JAXB reference implementation.

ed.mooney@sun.com

Joseph Fialli is a senior staff engineer at Sun Microsystems. He is currently working on the next version of JAXB. Previously within Sun, he was the specification lead for JAXB v1.0, lead architect for Java Message Service API v1.0.2, and added enhancements to Java serialization within J2SE 1.2.

joseph.fialli@sun.com



MORGAN KAUFMANN PUBLISHERS
AN IMPRINT OF ELSEVIER
WWW.MKP.COM

LEVERAGING TECHNOLOGY



Java Web Services Architecture
by James McGovern, Sameer Tyagi,
Michael Stevens, and Sunil Mathew

Java:
Practical Guide for Programmers
by Michael Sikora

The Struts Framework:
Practical Guide for Java Programmers
by Sue Spielman

JDBC:
Practical Guide for Java Programmers
by Gregory D. Speegle

Available This Summer

Unit Testing in Java:
How Tests Drive the Code
by Johannes Link and Peter Fröhlich

ORDERING INFORMATION

Elsevier Science
Order Fulfillment Dept.
11830 Westline Drive
St. Louis, MO 63146

Phone: (800) 545-2522
Fax: (800) 535-9935

Email: custserv.mkp@elsevier.com

“XML data binding relieves the pain of any Java programmer who has ever winced at having to work with a document-centric processing model”

XML documents, specifications, and documentation. It's perfect for experimenting with UBL applications. We're going to do just that using Java bindings generated by JAXB from the UBL schema.

Compiling the UBL Schema into Java Using JAXB

To compile the UBL schema into Java classes, download the Java Web Services Developer Pack 1.2 (see References). The Java WSDP is a free, integrated toolkit that allows Java developers to build, test, and deploy XML applications, Web services, and Web applications.

Set JWSDP_HOME to the directory where you installed the Java WSDP. Change to \$JWSDP_HOME/jaxb/samples/ubl. Create a directory named test, then run (xjc is JAXB's schema compiler):

```
$JWSDP_HOME/jaxb/bin/xjc.sh -d test 0p70/xsd/*.xsd
```

(You can also invoke xjc by way of an Ant task.) This creates the packages in test (see Listing 1). (Listings 1–7 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

By default, xjc puts its output into Java packages with names derived from the schema's targetNamespace.

Customizing the Package Names

JAXB gives you control over the binding process with various customizations. These can be inline in the schema by way of XSD annotation elements, or can be put in their own file. In the latter case, JAXB's xjc uses XPath to identify which part of the schema the customization affects. For example, this customization:

```
<jaxb:bindings
  schemaLocation="0p70/xsd/CoreComponentParameters.xsd"
  node="/xsd:schema">
  <jaxb:schemaBindings>
    <jaxb:package
      name="org.oasis.ubl.corecomponentparameters"/>
  </jaxb:schemaBindings>
</jaxb:bindings>
```

JAXB Package Name Customization

puts the Java classes and interfaces created from CoreComponentParameters.xsd in the org.oasis.ubl.corecomponentparameters package.

ubl.xjb contains this customization and

similar ones for each of the UBL schemas. External customization is useful for customizing standard schemas. It allows customization of schemas that are considered read-only. It also allows groups sharing a schema to bind to Java differently based on the needs of each application.

Create a directory named classes. Then compile the schema like this:

```
$JWSDP_HOME/jaxb/bin/xjc.sh -b ubl.xjb -d classes
0p70/xsd/*.xsd
```

This creates the packages in classes shown in Listing 2. See Chapter 10 of the *Java Web Services Tutorial* for more JAXB binding customizations.

A Typical Binding

UBL_Library_0p70_Reusable.xsd defines a number of types used throughout the UBL schema. For example, Listing 3 is a simplification of AddressType.

JAXB binds AddressType to the Java interface shown in Listing 4. You'll find this interface in the directory:

```
classes/org/oasis/ubl/corecomponentparameters
```

This class file contains its implementation.

xjc: The JAXB Binding Compiler

The Java WSDP comes with xjc.sh and an xjc.bat:

```
$JWSDP_HOME/jaxb/bin/xjc.sh -help
Usage: xjc [-options ...] <schema>
Options:
-nv          : don't validate the input schema
-extension  : allow vendor extensions
-b <file>   : external bindings file
-d <dir>    : output generated files to <dir>
-p <pkg>    : target package
-host <proxyHost> : set http.proxyHost
-port <proxyPort> : set http.proxyPort
-classpath <arg> : where to find user class files
-help       : display this help message
```

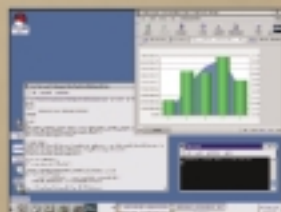
The build.xml sets up the xjc target like this:

```
<taskdef name="xjc"
  classname="com.sun.tools.xjc.XJCTask">
  <classpath refid="classpath"/>
</taskdef>
```


Throwing Our Hat In The Ring...



...with a whole new brew of Chart FX.



10 Years of creating quality developer charting solutions gives us an edge over others in the industry. So, although we're the new kid on this block, we've been in the neighborhood for a long time. The maturity of Chart FX and our support will prove it. Chart FX for Java is a 100% Java component that uses your J2EE application server and JSP technology to produce charts in a variety of formats: PNG, JPEG, SVG and Flash. Additionally, it will produce native Windows output in the form of ActiveX or .NET. Server-side Bean or Enterprise Java Bean (EJB). Data population through JDBC, XML, text files and more. *For more information or to download a trial version, visit www.softwarefx.com.*



Chart FX for JAVA

Learn more at
www.softwarefx.com

©2001 Software FX. All rights reserved. Chart FX is a registered trademark of Software FX, Inc. All other brands are owned by their respective owners.

Win a 2003 Honda Shadow Sabre!
< visit www.softwarefx.com to enter - no purchase necessary >




```
classes/org/oasis/ubl/commonagg:egstetypes/impl/
AddressTypeImpl.java
```

Compiling the Generated Code

It's easiest just to use Ant:

```
$JWSDP_HOME/apache-ant/bin/ant compile-generated
```

Creating the Javadoc

```
$JWSDP_HOME/apache-ant/bin/ant javadoc
```

JAXB's Javadoc is particularly useful since the generated interfaces contain Javadoc comments with the snippet of schema that's bound to them. You can use JAXB's Javadoc customization to add additional documentation.

The PrintOrder Application

The UBL sample in Java WSDP 1.2 reads in a UBL order instance, computes subtotals and a grand total, and prints the result to the screen. Sounds simple enough, but there's a lot going on. Here's how you run it:

```
$JWSDP_HOME/apache-ant/bin/ant printorder
```

Joese Office Supply 32 W. Lakeshore Dr Chicago, IL

Date: February 2, 2003

Sold To: George Tirebiter
c/o Bills Microdevices
413 Spring St
Elgin, IL 60123

- Part No.: 32145-12
Description: Pencils, box #2 red
Price: \$2.50
Qty: 5
Subtotal: \$12.50
- Part No.: 78-697-24
Description: Xerox Paper- case
Price: \$30.00
Qty: 12
Subtotal: \$360.00
- Part No.: 091356-3
Description: Pens, box, blue finepoint
Price: \$5.00
Qty: 10
Subtotal: \$50.00
- Part No.: 543-165-1
Description: Tape, 1in case
Price: \$12.50
Qty: 3
Subtotal: \$37.50
- Part No.: 984567-12
Description: Staples, wire, box
Price: \$1.00
Qty: 10
Subtotal: \$10.00
- Part No.: 091344-5
Description: Pens, box red felt tip
Price: \$5.00
Qty: 5
Subtotal: \$25.00
- Part No.: 21457-3
Description: Mousepad, blue
Price: \$0.50
Qty: 12
Subtotal: \$6.00

Total: \$501.00

Figure 1

This prints the itemized order to the screen, with subtotals for each item and a grand total for the order (see Figure 1).

PrintOrder contains the main() (see Listing 5). It relies on three facade classes:

- package samples.ubl.report
- import samples.ubl.report.facade. OrderFacade;
- import samples.ubl.report.facade. OrderLineTypeFacade;
- import samples.ubl.report.facade. AddressFacade;

The Facade design pattern provides a simpler, higher-level interface. Typically, that's done to isolate an application from the complexity of the underlying subsystems. Here, our principle motivation is to isolate our application from changes to an evolving schema. Since we just wanted to print a simple report, our facades are read-only.

A look at printBuyer() in PrintOrder shows that getting the name of the

buyer contact (a person, usually) is just a matter of calling OrderFacade's getBuyerContact() method (see Listing 6).

getBuyerContact() has a simple signature (see Listing 7). In this listing, order is a reference to org.oasis.ubl.order.Order, which is bound to <xsd:element name="Order" type="OrderType"/> in UBL_Library_0p70_Order.xsd. If that binding were to change, for example, such that order.getBuyerParty() returned PartyType, we could make this one-line change to getBuyerContact() without affecting PrintOrder:

```
(BuyerPartyType)party = order.getBuyerParty();
```

This pattern characterizes the other methods in OrderFacade and the methods in OrderLineTypeFacade and AddressFacade. And none of this code relies on org.w3c.dom or org.xml.sax interfaces.

Conclusion

JAXB makes it possible to process XML documents without using SAX or DOM. This saves us time when getting started and over the long haul since we don't have to debug and maintain a lot of tedious code devoted to traditional XML processing. While this is useful for any application that uses XML, it will be especially valuable in those application domains with defined XML Schemas for business data interactions. ☺

References

- **UBL0p70.zip:** <http://oasis-open.org/committees/ubl/lcsc/0p70/UBL0p70.zip>
- **UBL Technical Committee:** www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl
- **JSR 31:** <http://jcp.org/en/jsr/detail?id=31>
- **Java Community Process:** <http://jcp.org/>
- **JAXB home page:** <http://java.sun.com/xml/jaxb/index.html>
- **Java Web Services Developers Pack 2.0:** <http://java.sun.com/webservices/webservicespack.html>
- **Java Web Services Tutorial:** <http://java.sun.com/webservices/docs/1.1/tutorial/doc/index.html>
- Gamma, E., Helm, R., Johnson, R., and Vlissides, JM. (1998). *Design Patterns CD: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Resolving Name Collisions

Since each XML namespace contains six symbol spaces, it's not uncommon for terminals in one symbol space to collide with those in another when JAXB tries to map them to Java. This inlined customization, which handles them automatically, illustrates the technique:

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
.
<xsd:annotation>
<xsd:appinfo>
<jxb:schemaBindings>
<jxb:nameXmlTransform >
<jxb:elementName suffix="Element"/>
</jxb:nameXmlTransform >
</jxb:schemaBindings>
<xsd:appinfo>
</xsd:annotation>
.
</xsd:schema>
```

Customizing Names of Classes and Properties

The text explains how we used an external binding customization to change the name of the packages used for the generated Java code. JAXB also lets you rename classes and properties.

Setting Global Binding Behavior

This illustrates only some of the possible global bindings:

```
<globalBindings collectionType="indexed"
fixedAttributeAsConstantProperty="true"
generateIsSetMethod="true"/>
```

This customization:

- Represents collections as a JavaBean indexed property that also has a length() method (by default, JAXB collection properties are java.util.List).
- Maps all fixed attributes as Java constants.
- Generates an isSet method for JAXB properties with a primitive base type (such as int) or base type of List. The isSet method allows you to determine if the getter for an optional property is returning the property's default value or if it is returning a value set within the XML document. This helps you determine what was in the original XML document, although typically, JavaBeans don't care.

Controlling the Binding of an XML Schema Element to its Java Representation as a Type-Safe Enumeration Class

```
<jxb:globalBinding
```

typesafeEnumBase="xsd:string"/> can be used to customize the enumeration members and their values of simpleType with enumeration facets that derive by default from xsd:string (there's a default type-safe enumeration binding for NCName).



Add Presence and Instant Messaging features to your software with JBuddy SDK™ and start seeing the big picture.

Customers have more expectations than ever before when it comes to your products. With JBuddy SDK, your software will deliver new enterprise-class features including remote application access, real-time messaging, and presence; features already familiar to your customers. You'll save precious development time and resources while adding real value to your applications. See how simple connecting your products to public and enterprise Instant Messaging networks can be with JBuddy SDK.

For more information and to request a FREE 30 day evaluation download, visit our website at www.zionsoftware.com/products/jbuddy

 ZION SOFTWARE
www.zionsoftware.com

Real Solutions for the
Real-time Enterprise.™

An Empirical Study

Microtuning your application
by Haralambos Marmanis

Micro-Performance

This article

addresses the performance of Java applica-

tions with respect to the underlying JVM, and is based only on the standard API classes and algorithms that are implemented by them. We do this in order to establish results that are widespread in their applicability. No matter what code you write, you can decompose it into parts that can be studied as individual units. The performance of the whole is equal to the performance of its parts plus the overhead of the interaction between the parts. The simplest parts that you can decompose are the classes that are offered by the Java API. Thus, knowing how well these parts perform can be crucial to the overall performance of your product.

Before we go any further, I'll justify why it's a good idea to know about micro-performance. It may be considered unnecessary to examine the performance at its finest granularity. However, most performance-tuning strategies neglect the fact that the right choice, at the micro level of the code, can squeeze out speed without making the code more complicated or error prone. Moreover, it is something that everyone can do; no special training is needed to choose the method that is the fastest of a variety of possible methods and equally effective for the task at hand.

Typically, an engineering team will employ a profiling tool that will pinpoint the location of "hotspots" (memory- and/or CPU-intensive code fragments). This is certainly a valid way of improving performance, but it says nothing about global performance. If your code throughout is slow, then a profiling tool won't help.

Global performance excellence stems from a global enforcement of best practices, i.e., fast implementations throughout the code. We would like to see extremely fast Java applications, especially enterprise applications, and our maxim is that optimal code should be used everywhere in the source code. A fitting analogy here is the saying, "A water tower can be filled one teaspoonful at a time." Best practices can be revealed by a detailed study of the available APIs and documentation of the findings. Implementing the fastest code doesn't necessarily mean the code will be more error prone; you can be fast and accurate simultaneously. I hope this article contributes toward that end.

For the purpose of illustrating micro-performance benchmarks, we will use four Java Virtual Machines. Three of them (J2SE 1.3.1_07-b02, 1.4.0_03-b04, and 1.4.1-b21) are provided

How fast should our Java code be to be considered fast? After all, speed is a relative concept. I'll compare the results of CPU performance for the following JVMs: Sun's J2SE 1.4.1, 1.4.0, 1.3.1, and Jikes. These results can be used to make a number of educated decisions such as choosing a JVM, deciding on algorithmic designs, and selecting the right method from the API. They provide an overall assessment of performance that's not custom related since the code used is quite common and drawn directly from Sun's Java APIs.

This article studies the Java APIs for an extra boost in performance. It's not a new idea, and is often referred to as micro-benchmarking (MBM). However, a systematic and thorough performance analysis at that level is still missing. Herein, I'll address performance as speed, measured in wall clock time. I'll cover various Java Virtual Machines (JVMs), and show that the results differ significantly. A study of memory consumption is also warranted, but it will not be addressed here; for such an analysis, visit www.marmanis.com. See the Resources section for references to performance studies.

When dealing with performance, one of the major difficulties is the many "scales" or layers that are usually involved in Java applications, especially in enterprise Java applications. I categorize performance problems based on their scale:

1. System architecture
2. Algorithm selection
3. Code implementation
4. System configuration
5. System infrastructure

Only categories 1, 2, and 3 are directly related to the Java programming language. Problems in category 1 can be dealt with or, even better, prevented, by the proper use of J2EE Blueprints, the Java version of design patterns for enterprise applications (see <http://java.sun.com/blueprints/enterprise> and www.theserverside.com/patterns/index.jsp). Categories 4 and 5 involve handling and testing components that may be irrelevant to Java per se. These categories are given in order of decreasing importance. Experience shows that you will get most of your performance increase from improvements in categories 1 and 2, regardless of how you measure performance! Nevertheless, if you want to squeeze as much performance as possible out of your infrastructure, it's worth knowing what the performance of your fundamental APIs is. This part of performance is aptly called micro-performance and it belongs to category 3.

THE FASTEST EJB APPLICATION DEVELOPMENT CYCLE ON THE PLANET.



Ensemble Glider's EJB container simulator provides the world's fastest and easiest way to run and debug EJB applications.

Download a free trial today at

www.ensemble-systems.com/glider



ensemble

www.ensemble-systems.com

by Sun Microsystems, Inc. (<http://java.sun.com/j2se/>), and the other (Jikes 1.3.0) is provided by IBM (<http://oss.software.ibm.com/developerworks/opensource/jikes/>). One of the design guidelines for Sun's version 1.4 was to improve the performance and scalability of the Java platform; you can read more about their specific rationale at <http://java.sun.com/j2se/1.4/performance.guide.html>.

The bytecode for each run was created by the compiler that comes with each distribution. (The source code can be downloaded from www.sys-con.com/java/sourcec.cfm.) All the compilers were invoked as follows (see also the scripts that are provided):

```
%JVM_HOME%\bin\javac -g:none -O Test[i].jaa
```

where %JVM_HOME% is the path to the distribution that we target, and Test[i].java is the Java class that corresponds to one of our tests (e.g., Test2.java). The flag -O eliminates optional tables in the class files, such as line number and local

offer the following options: client, server, and hotspot.

For the 1.4.x versions, the hotspot is a synonym for the client JVM. I've chosen to use the server JVM, although it should be easy for the user to experiment with the client JVM by changing the respective flag in the scripts. In general, the differences between the server and the client versions are related to the JVM tuning. The client JVM is tuned to reduce application startup time and memory footprint, which is important when running desktop applications. The server JVM is intended for use in server applications where the JVM will run for long times and peak performance is more important than footprint and rapid startup. Both options are of interest, although Java is clearly more prevalent on the server side.

Last, I've also chosen to fix the size of the heap in order to remove the burden of resizing the memory, which is one of the garbage collection responsibilities. This doesn't prohibit the garbage collector from doing its work. The question that we really ask is this: Given a fixed amount of memory for each JVM, which JVM performs the exact same code faster?

"Global performance excellence stems from a global enforcement of best practices, i.e., fast implementations throughout the code"

variable tables. This provides only a small performance improvement on the generated code, although if our class files had been sent across a network, it could have helped significantly. IBM and Sun have no plans for bytecode optimization; they'd rather focus on runtime optimization (see www.nejug.org/2000/sept00_slides/java_perf.htm).

Once the bytecodes were created, they were executed by the target runtime:

```
%JVM_HOME%\bin\java -server %JVM_XMX% %JVM_XMS% Test[i]
```

where %JVM_HOME% is again the path to the distribution that we target, and Test[i] is the Java bytecode that corresponds to one of our tests (e.g., Test2.class). The JVMs by Sun

During that time – not longer than a few minutes in the worst case – the JVM that spends the least time dealing with garbage collection will have an advantage over the other JVMs.

I'll employ only one operating system platform, namely, the Windows 2000 Professional. However, it should be clear that for a complete and useful assessment of micro-performance, the same benchmarks should be run for other operating systems as well, such as Linux, Solaris, and AIX. The Windows system that I'll use runs on a Dell Inspiron 4100, with total physical memory of 654,776KB; BIOS PLUS Version 1.10 A09; and x86 Family 6 Model 11 Intel CPU at 1,100MHz. Results for a Linux system that runs on a Micro PC, with total physical memory of 772,856KB and an AMD Athlon Processor at 1,134MHz, should be available on my Web site. For the Linux platform, there is also a JVM that's offered by the Blackdown open source project (see www.blackdown.org). There are more JVM implementations available and I'll make an effort to include as many of them as is possible on my Web site.

The Benchmarks

I'll present 19 benchmarks that cover some of the following: basic arithmetic operations, java.lang package, java.io package, java.util package, and java.security package. I refer to each test by concatenating the character "T" and the respective enumeration of the test. Thus T1 will refer to Test 1, T2 will refer to Test 2, and so on. Obviously, this is not an exhaustive list and the choices are based on what I think are popular method calls.

The code for the benchmarks was written with simplicity in mind. The theme is the same for all benchmarks. They all consist of some setup code and some code inside a for loop whose timing is the goal of each benchmark. Thus, each benchmark repeats for a "reasonable" time a call to a small piece of Java code. I use System.currentTimeMillis() to measure the wall clock time (in ms). To take into consideration the time spent

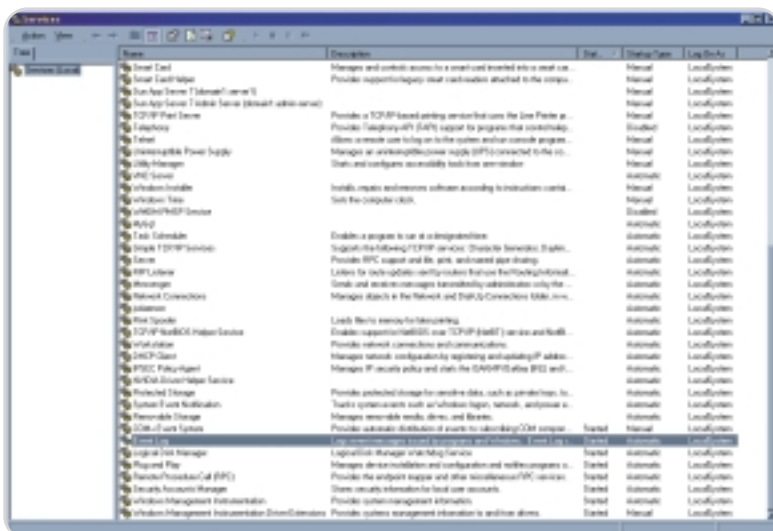


Figure 1 List of services

Visit us at
JavaOne
Booth
#1336



Need an Embeddable Process Engine Without All the BAGGAGE?

We've got
The AnswerSM

Let's get right to the point. **Reactor 5** from Oak Grove Systems is the process execution framework that gives you exactly what you need. Period.

One lean, mean, singularly focused machine, **Reactor 5** is already the solution of choice for such firms as SAS, Sybase and Plumtree Software, allowing them to cut their costs, speed their time to market and free their developers to concentrate on their core products. **Reactor 5's** benefits include:

- **Universal Deployability** - Reactor 5 is compatible across all J2EE application servers
- **Easy Integration** - Featuring an open, standards-based architecture
- **The Only Process Engine You Need** - For workflow automation, business process management and Web Services orchestration

Available as source code with royalty free distribution!

So take a load off. And you can start by calling us at 1-818-880-8769, or by **downloading your free evaluation copy** at www.oakgrovesystems.com/jdj

oakgrove
systems

The AnswerSM - For All Your Process Execution Needs

Workflow Automation – Business Process Integration – Web Services Orchestration

for the loop, I always run a baseline test first to obtain the reference time, i.e., the time spent in a loop without code in it. Rather than subtracting the reference time from the reported value, I report both. I'll also report the number of iterations since this varies among our benchmark tests. I'll often include several method calls inside the loop so we can examine the efficacy of some API classes in an aggregate fashion; see, for example, the benchmark Test5.java. A more granular approach is, of course, what this article proposes and I'll publish and maintain on the Web more fine-grained results.

Let's now see what each test measures and analyze its results. All values refer to ms and the loop size was chosen so that I'd weed out any fluctuations of the CPU due to unrelated

"In the competitive market of enterprise apps, it's worthwhile to get as much performance as you can out of the standard APIs"

processes. Figure 1 shows a snapshot of the list of services that run during the test and Figure 2 shows a snapshot of the process list from the TaskManager of the Windows OS. Inside parentheses I include the ratio of performance for each JVM when compared to the Sun 1.4.1 JVM. Hence, a value smaller than 1 means that the JVM is faster than Sun's 1.4.1 JVM, and a value larger than 1 means that the JVM is slower than Sun's 1.4.1 JVM; therefore the value inside the parentheses will always be equal to 1 for the last column.

1. T1 measures the performance for typical numerical operations. I use a long and two double numbers, and perform an addition, a multiplication, and a division with constant numbers.
2. T2 defines a number of variables of type string and uses the method equals to compare them. Inside the loop I use several different string comparisons, since the speed of the algorithm is not uniform across all possible strings. Hence, my results will give a good estimate of the method's performance for strings that are equal in length or vary by one character.
3. T3 has the same setup as T2 but uses the method equalsIgnoreCase.

4. T4 has the same setup as T2 and T3 but uses the method compareTo.
5. T5 tests the performance of some commonly used mathematical functions. The class java.lang.Math has various useful mathematical functions. We test the method that creates a random number, random(); the method that calculates the cosine of an angle, cos; the method round that returns the closest long to its argument of type double; the method that calculates the absolute value of a number, abs; the methods that return the exponential and the logarithm of an argument, exp and log, respectively; the method that gives us the maximum between two numbers, max; the method that gives us the square root of a number, sqrt; and

finally the method that raises one number to a certain power, pow.

6. Another common task for many applications is the reading of a properties file. This is usually done by reading the file via the java.io API and loading the values on a Properties class that's provided in the java.util package. I do just that in T6.
7. As you may very well know by now, string concatenation is much slower than the append method of a StringBuffer. T7 tests how quickly this works for the various JVMs under study. Some time is spent to find the length of the StringBuffer each time and delete all its content so that the StringBuffer is empty at the beginning of each iteration. In total, we have six append calls, one length, and one delete.
8. In T8 I measure the performance of the various classes that are needed to encrypt and decrypt a 128-character string. I use the SunJCE provider and a triple DES key spec. In particular, I initialize the Cipher with "DESede/ECB/PKCS5Padding." Jikes does not run with exactly the same code in this case, so an N/A appears in the corresponding position of the table.
9. In T9 I serialize, write to the disk, read from the disk, and deserialize a Vector object.
10. In T10 I test the performance of the method add in an ArrayList.
11. In T11 I test the performance of the method add in a Vector.
12. In T12 I test the performance of the method add in a HashSet.
13. Since the above three tests add Random objects in the various collections, I run a test, (T13), that measures the time that the generation of these objects takes. These times will be referenced side-by-side with the times that I obtain from T10, T11, and T12.
14. In T14 I test the performance of the method remove in an ArrayList.
15. In T15 I test the performance of the method remove in a Vector.
16. In T16 I test the performance of the method remove in a HashSet.
17. A typical way of iterating through the elements of a col-

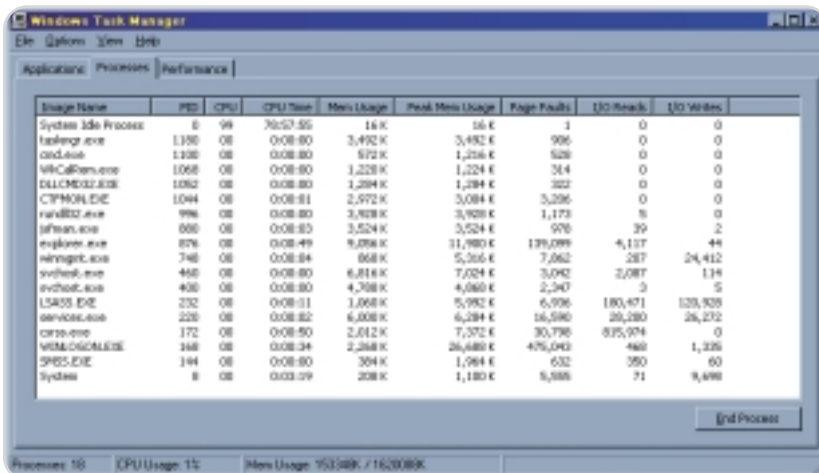


Figure 2 Process list

Embed powerful reporting functionality in your servlets, JSP, & Java applications with the Formula One Reporting Engines!



Formula One e.Report Engine

For PDF, XML, HTML, DHTML and email reports.

Free Trials & Demos: www.ReportingEngines.com/info/JDJ_June_ere.jsp

Formula One e.Spreadsheet Engine

For richly formatted Excel reports.

Free Trials & Demos: www.ReportingEngines.com/info/JDJ_June_esc.jsp



888-884-8665 • 913-851-2200
sales@ReportingEngines.com



lection is by using the Iterator object. In T17, I obtain the Iterator of an ArrayList and iterate through all its elements.

18. In T18 I obtain the Iterator of a Vector and iterate through all its elements.
19. In T19 I obtain the Iterator of a HashSet and iterate through all its elements.

Conclusion

I cannot emphasize enough the importance of the data on the performance of a method call when attempting to do micro-benchmarks. The space that is usually covered by the arguments of an operation, or of a method call, is vast. Although indicative values can be obtained, the results are

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T1	50,000,000	100 (10)	20 (2)	20 (2)	10 (1)

Table 1

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T2	50,000,000	15723 (0.88)	8462 (0.47)	16063 (0.90)	17806 (1)
T3	50,000,000	122646 (1.26)	146551 (1.51)	99403 (1.02)	96989 (1)
T4	50,000,000	33027 (412.8)	100 (1.25)	80 (1)	80 (1)

Table 2

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T5	50,000,000	5928 (0.05)	122787 (1.02)	134934 (1.12)	120553 (1)
T6	2,000	2764 (1.22)	2329 (1.03)	2136 (0.94)	2270 (1)
T7	50,000,000	32046 (0.98)	32527 (0.99)	31635 (0.97)	32737 (1)

Table 3

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T8	100,000	n/a	178606 (5.06)	58815 (1.67)	35281 (1)

Table 4

applicable only for the space of the arguments that they cover. With that "alert" status in mind, let's proceed and draw some conclusions.

The results for T1 (see Table 1) show that the Sun JVMs take advantage of the fact that the second operand is a constant and achieves some aggressive optimization of arithmetic operations. Nevertheless, all JVMs are quite fast and achieve billions of operations per second.

The results of T2, T3, and T4 (see Table 2) show that compareTo is two to three orders of magnitude faster than either of the equals methods. In my study – not shown here – I have found that the method call equalsIgnoreCase of the class String is a lot faster than the method call equals of the same class when the majority of the compared data have different lengths. There is a good reason for this, of course. The equalsIgnoreCase first checks the length of the two strings. Nevertheless, the point is that if you know that piece of information and you happened to extensively use string comparisons, you can take advantage of it without paying a penalty; if case does matter, then obviously this is not appropriate!

The results of T5 (see Table 3) show that the mathematical functions are two orders of magnitude faster with Jikes than with any of the Sun JVMs. Hence, if you rely heavily on computing cosines and logarithms, you should definitely take that into consideration when picking your JVM. However, the results of T6 and T7 show that there is not really a difference between the JVMs when it comes to loading a properties file and using the append method, respectively.

The results of T8 (see Table 4) show that encryption-related methods in the Sun 1.4.x JVMs are an order of magnitude faster than the same methods in Sun 1.3.1. Thus, if encrypting and decrypting is bread and butter for you, you have one more reason to upgrade your JVM!

The results of T9 (see Table 5) show that serialization to (upper readings) and deserialization from (lower readings) a file with the Sun 1.3.1 JVM is faster than any other JVM. Jikes is faster than both of the Sun 1.4.x JVMs. However, all the JVMs are in the same order of magnitude in terms of the time spent to accomplish the task.

The results of T10, T11, and T12 (see Table 6) show that the add method is equally fast for an ArrayList and a Vector. However, Jikes is faster by a factor of at least two, regardless of the Collection class that's used. A somewhat disturbing result is that the Sun 1.4.x JVMs seem to be slower than the Sun 1.3.1 JVM for the ArrayList and the Vector classes. This result consistently appeared in the runs that were made in preparation for this article, so there should be a reason for it. As we'll see later, that doesn't happen with the remove method or the iterator. It would be nice if the Sun engineers would take a look at it.

As expected, the add method in the HashSet class is slower than the same method for the ArrayList and the Vector classes. It's extremely slow in Sun 1.3.1, by three orders of magnitude when compared to all other JVMs. The removal and the iteration in the HashSet class are also slow with the Sun 1.3.1 JVM, by an order of magnitude compared to all the other JVMs.

The results of T14 and T15 (see Table 7) show that the remove method is equally fast regardless of the JVM. T16 shows that the same method is faster for a HashSet than for an ArrayList or a Vector; however, the method is an order of magnitude slower among the tested JVMs for the Sun 1.3.1 JVM.

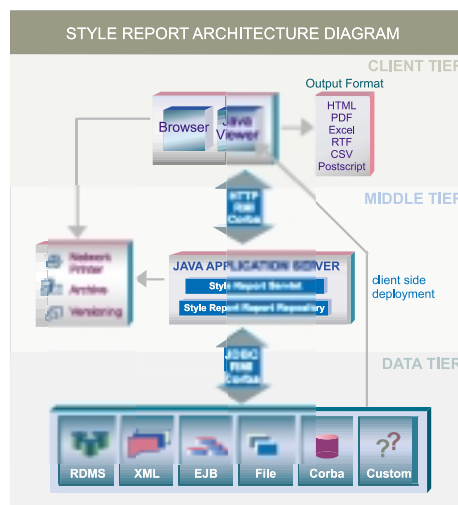
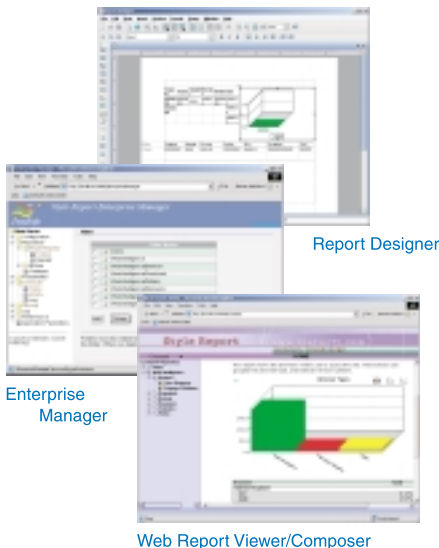
Finally, the results of T17, T18, and T19 (see Table 8) show that the iteration with the Sun JVMs is faster than the iteration with Jikes by, at least, a factor of two; the caveat here is

Java & Web Reporting: with Java flexibility without Java complexity

Style Report: Information Vantage Point



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



Style Report Features

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

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T9	1,000,000	5678 (0.87)	4687 (0.72)	9344 (1.43)	6549 (1)
		4978 (0.86)	4035 (0.70)	5738 (1)	5758 (1)

Table 5

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T10	4,000,000	2364 (0.31)	4106 (0.54)	8062 (1.06)	7621 (1)
T11	4,000,000	2413 (0.31)	4146 (0.54)	8072 (1.05)	7671 (1)
T12	1,000,000	1502 (0.34)	250089 (57.28)	4576 (1.05)	4366 (1)
T13	4,000,000	2003 (0.46)	2313 (0.53)	4787 (1.09)	4396 (1)
	1,000,000	511 (0.43)	641 (0.54)	1292 (1.08)	1192 (1)

Table 6

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T14	100,000	6759 (1)	6760 (1)	6749 (1)	6760 (1)
T15	100,000	6750 (1)	6760 (1)	6760 (1)	6759 (1)
T16	100,000	30 (0.43)	281 (4.01)	71 (1)	70 (1)

Table 7

Test	Loop size	Jikes Time	Sun131 Time	Sun140 Time	Sun141 Time
T17	4,000,000	821 (3.04)	320 (1.19)	230 (0.85)	270 (1)
T18	4,000,000	891 (2.87)	401 (1.29)	360 (1.16)	310 (1)
T19	4,000,000	321 (1.15)	3045 (10.88)	310 (1.11)	280 (1)

Table 8

the problematic iteration of the HashSet, mentioned earlier.

The study presented here should not be considered complete. The purpose of this article is to distinguish the many scales that may affect the performance of Java applications, pay particular attention to what I call micro-performance, and to suggest one way to tackle the problem through detailed benchmarks of the APIs. The article demonstrates these ideas by employing a very small, but quite popular, portion of the APIs.

The performance of the Java API method calls clearly depends on the JVM. However, the differences are not uniform – you can't claim that JVM-1 is always faster than JVM-2. That in itself is not news, of course, but it is important to quantify the differences because you may find that for your own application JVM-1 is better than JVM-2, and you may want to instill micro-benchmarking in your own or your team's coding practice.

In the competitive market of enterprise applications, it's worthwhile to get as much performance as you can out of the standard APIs. To know how to get that advantage, we need to quantify the performance of the Java language at the level of its APIs. The point is that if you can get faster code without "collateral damage" – to use a, regrettably, quite fashionable term at the time of this writing – why not do it?

It was my initial desire to include the analysis of the GC output in this article; however, this would double, if not triple, its size. Nonetheless, I strongly recommend you collect such output (e.g., by using `-verbose:gc` as a flag to the JVM) and observe how the various JVMs collect their garbage over time. That's quite instructive. In addition, if you feel the urge to truly understand your JVM, use its optional flags as new parameters in your analysis of the results.

For any application of substantial size and complexity, a

proper micro-performance tuning may produce significant speedups. Not comparable to the speedups that you can get by choosing a better architecture, or a better algorithmic approach; at the first stages of performance tuning even a factor of 10 is possible, in some cases. But when your architecture is appropriate and your algorithms optimal, it is likely that micro-tuning your application is a nice-to-have weapon in your arsenal. If you have already adopted micro-tuning during your code implementation, you're probably grinning right now in a self-satisfied manner! ☺

Resources

- Shirazi, J. (2003). *Java Performance Tuning, 2nd Edition*. O'Reilly & Associates, Inc.
- Wilson, S., and Kesselman, J. (2000). *Java Platform Performance: Strategies and Tactics*. Addison-Wesley Pub Co. <http://java.sun.com/docs/books/performance/>
- *BEA WebLogic JRockit Virtual Machine*: www.bea.com/products/weblogic/jrockit/index.shtml
- *Java 2 Platform, Standard Edition (J2SE)*: <http://java.sun.com/j2se/>
- *Jikes*: <http://oss.software.ibm.com/developerworks/opensource/jikes/>
- *Blackdown Project*: www.blackdown.org
- *Java BluePrints*: <http://java.sun.com/blueprints/enterprise/>
- *TheServerSide.com*: www.theserverside.com/patterns/index.jsp
- *Java 2 Platform, Standard Edition (J2SE)*: http://java.sun.com/j2se/1.4/performance_guide.html
- Hagggar, P. "Improving Java Code Performance": www.nejug.org/2000/sept00_slides/javaperf.htm



Haralambos Marmanis is a software architect at Zeborg. He has more than 12 years of software development experience in academia and the industry. He received his PhD in applied mathematics and scientific computing from Brown University. His interest is in multitier, high performance, enterprise software.

marmanis@computer.org

WebAppCabaretTM

www.webappcabaret.com

**J2EE Web Hosting****JavaOne
SPECIALS**

Quality Web Hosting at a reasonable price...

◀ JAVA WEB HOSTING AND OUTSOURCING ▶

You have developed the coolest mission-critical application. Now you need to deploy it.

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

OUTSOURCING:

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

J2EE HOSTING:

Below is a partial price list of our standard hosting plans. (**Reseller accounts also available**). For more details and for at least \$25 off the total price of your order please log on to <http://www.webappcabaret.com/javaone.jsp>

Guaranteed Latest Tomcat/JBoss/Jetty for New Enterprise Accounts

Features	Basic	Professional	Enterprise	Dedicated
Monthly Cost	us\$9.00	us\$17.00	us\$39.00	us\$191.00
Servlet Spec	2.3	2.3	Latest	Latest
JSP Spec	1.2	1.2	Latest	Latest
EJB Spec	1.1	1.1	Latest	Latest
Private JVM	x	x	x	x
Database	1	1	5	Unlimited
e-Commerce		x	x	x
Tier 1 Center	x	x	x	x
Accounts/Server	120	120	35	1
RAM/Server	2GB	2GB	2GB	256MB
RAID Diskspace	30MB	200MB	900MB	40GB
Bandwidth/Month	3GB	10GB	15GB	35GB
Backup	x	x	x	x
Domains	1	2	5	Unlimited
Email Accounts	2	20	100	Unlimited
Servlet Contexts	1	1	10	Unlimited
Control Panel	x	x	x	x
FTP	x	x	x	x
Telnet/SSH		x	x	x
Web Mail	x	x	x	x
Web Stats		x	x	x
Perl/PHP		x	x	x
Dedicated Apache			x	x
Engine Choices	x	x	x	x
WAR/EAR	x	x	x	x
Tomcat	4.0.3	4.0.3	Latest	Latest
JBoss	2.4.1	2.4.1	Latest	Latest
Jetty			Latest	Latest

Prices, Plans, and Terms subject to change without notice. Please log on to our website for the latest Prices and Terms.
Copyright © 1999-2003 WebAppShowcase • All rights reserved • Various trademarks held by their respective owners.



Glen Cordrey
J2ME Editor

Interesting Technologies

A recent J2ME-related announcement I find particularly interesting for a number of reasons is esmertec's demonstration of a MIDP 2.0 implementation on BREW. First, esmertec recently acquired Insignia, whose Jeode Embedded Virtual Machine for Java has made PersonalJava available on PDAs for some time. Is this acquisition the beginning of consolidation in the J2ME world, or an isolated incident?

Second, this is the first deployment of a MIDP 2.0 implementation I've heard of. While MIDP 1.0 has been available on BREW via Insignia products for some time, the final MIDP 2.0 spec was released just this past December, and the short turnaround time for Insignia's implementing 2.0 is a statement of their confidence in J2ME.

And finally, the availability of the MIDP on BREW benefits both J2ME and BREW. While BREW and J2ME are sometimes characterized as competitors, there are many reasons to consider them as complementary technologies. J2ME on BREW makes a much larger developer community and, consequently, a much larger set of applications available to BREW-enabled handsets. Conversely, BREW provides J2ME developers with ready-made services for provisioning, billing, and revenue collection.

One concern with the MIDP on BREW is whether the layering of the MIDP on top of BREW will introduce a significant performance penalty. Of course, one approach for improving performance on any JVM is the use of software techniques such as just-in-time and ahead-of-time compilation, which are as applicable in the J2ME world as in the J2SE world. But handheld devices also lend themselves to performance improvement via hardware enhancements that may not be applicable to or as attractive for PCs and servers.

I recently spoke with Jerry Steach of NanoAmp about one such technology, their MOCA-J product. This is a combination of a dedicated Java bytecode accelerator (206 of the 227 Java bytecodes are executed directly in hardware rather than being interpreted) and a

cache. In addition to significantly faster execution of Java applications (NanoAmp claims a 20x improvement over the Sun reference KVM), hardware execution of bytecodes plus on-chip power management uses significantly less battery power than software execution of the bytecodes, and battery drain is always a concern with mobile devices.

The MOCA-J accelerator and cache are two separate dies bonded together that are in turn bonded to a flash memory die provided by a chip manufacturer. Since handsets already require memory, melding the MOCA-J with already-needed flash allows the incorporation of MOCA-J into the handset without using any additional circuit board space, also a significant concern with mobile devices.

For me one of the attractions of J2ME is the variety of technologies that come into play when considering the J2ME landscape. Interesting combinations such as BREW and J2ME, innovative hardware solutions (such as MOCA-J) to the resource limitations of mobile devices, and the incorporation of other technologies such as Bluetooth, all create a multi-hued palette that colors the J2ME landscape.

• • •

Here at **DDJ** the majority of our articles come from you, our readers. A reader will propose an article idea (via www.sys-con.com/java/authors), which we'll review for topicality and other factors. If the proposal is accepted, we then work with the author to scope the article size, review drafts, and move the article toward publication.

I ask you to consider sharing with our readers your experiences and knowledge gained from developing J2ME applications. It's a great way to make new contacts in the industry – after my first article I received e-mail from developers in Hong Kong and France – and having a “publications” item on your résumé can make it distinctive, which can be of particular value in today's tight job market. And there is always the personal satisfaction to be gained by attempting something different and out of the ordinary. ☺



J2ME Clients
with Jini Services **64**

Interesting Technologies

For me one of the attractions of J2ME is the variety of technologies that come into play when considering the J2ME landscape.

62

P800 by Sony Ericsson

For those of you who haven't already come across a review of Sony Ericsson's smart phone offering, I'll briefly run through what the P800 offers outside of the Java space

72

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

glencordrey@sys-con.com

BREW™ CAN TURN J2ME™ INTO \$4U.



GOT J2ME™ APPS? BREW™ = IMED8 OPR2NTY 4 JAVA® DEVELOPERS 2 BLD FORTUNE. In plain, simple English, BREW is the open, end-to-end wireless development solution that's compatible with Java. The BREW Distribution System can put your J2ME application into the hands of millions of paying customers fast. And the worldwide market is growing, as operators and OEMs continue to adopt the BREW platform. Learn how you can make money with BREW for Java applications. Read the *Brew and J2ME White Paper* at www.qualcomm.com/brew.



Customize. Personalize. Realize.™

J2ME

CLIENTS

WITH

JINI SERVICES

Develop a highly portable, resilient service-oriented architecture

by Nikhil Patil & Ron Dearing

Jini provides simple and reliable access to services over any network, independent of platform, protocol, or application technology. Enterprises can use Jini to develop a resilient service-oriented architecture that can be accessed from a broad range of clients.

Java 2 Micro Edition (J2ME) enables developers to build portable rich-client solutions that can operate in either connected or disconnected modes. The combination of J2ME-enabled devices interacting with enterprise systems through Jini services provides a flexible architecture for building highly reliable, end-to-end mobile solutions.

However, for any client to directly access and/or run Jini services, it must be capable of dynamically downloading and executing Java classes and be able to participate in Jini Discovery and Join protocols. Currently, many J2ME-enabled devices with limited resources don't have this capability and can't directly participate in a Jini network. This article provides an overview of the technologies, and, with the help of an example, illustrates how to overcome the limitations of J2ME to develop an effective mobile architecture based on J2ME and Jini.

Jini Primer

From the official Jini architecture specification, www.sun.com/software/jini/specs, the Jini system is defined as:

...a distributed system based on the idea of federating groups of users and the resources required by those users. The focus of the system is to make the network a more dynamic entity that better reflects the dynamic nature of the workgroup by enabling the ability to add and delete services flexibly.

A Jini system uses the network as a foundation to enable service discovery and service execution. Traditional systems attempt to mask the appearance of the network; in contrast, Jini uses the dynamic, flexible nature of the network to form communities and register, discover, and invoke services. Unlike traditional systems, Jini is founded on the assumptions that networks are unreliable, change frequently, and should not require substantial maintenance by an administrator.

To address the assumption that networks are unreliable, Jini systems are self-healing. Jini services repair themselves using the concept of leasing. Leasing ensures that Jini services are automatically removed from the community after a

specified time period, without the need for administrator intervention. The network topology can change, but Jini will continue to work without the need to update the URLs referenced in services, change properties files, or perform manual administration. Jini services adapt to such changes automatically. Communication with a Jini service occurs through service proxies that are downloaded to the client machine automatically without administrator involvement, a practice that is normally required in traditional architectures.

Jini Architecture

The Jini architecture (see Figure 1) is comprised of the following elements:

Jini Service

A Jini service is an entity that can be a hardware device or a software component that provides functions like printing a report, looking up a stock quote, or executing an algorithm. Applications and other Jini services can use a Java interface and a Java proxy class to access a Jini service. The Java interface provides the definition of the methods that are available from the Jini service. The Java proxy provides the communication channel between the client and the actual service. Jini services are registered with the lookup service and are capable of being invoked through their public interface, which is defined via a Java remote interface. In most cases, the underlying system that allows Jini services to communicate is RMI. A Jini service may also use protocols such as SOAP, XML/HTTP, or CORBA.

Lookup Service

The lookup service, which is a Jini service, keeps track of the Jini services and provides proxies to communicate with the services. Sun provides a lookup service called Reggie (used for developing this example) as part of the Jini development kit.

Jini Client

The Jini client is any software that requests the proxy from the lookup service in order to invoke the Jini service.

RestaurantFinder Application

Now that you have a good understanding of Jini and J2ME, let's move on to an example application that uses Jini and J2ME to provide a valuable service to mobile users. We'll use this example to show you how to develop a complete



J2ME



J2SE



J2EE



HOME

Fiorano

Integration Middleware

Your one stop solution center for Enterprise Application Integration (EAI), BPM, Distributed Workflow and Messaging Middleware.

FioranoMQ™ JMS Server - Enterprise Messaging Infrastructure	Tifosi ESB™ Enterprise Service Bus Mid Tier Integration Solution	Tifosi Enterprise Integrator™ Comprehensive full featured Integration Broker Suite
Fastest, Most Scalable JMS Server	Web Services Integration including support for SOAP, WSDL, UDDI	100+ Packaged and Legacy Application Adapters
24x7 Reliability and Hot Failover	Service Oriented Architecture	EAI, BPM and Distributed Workflow
Standards based	High Availability	Comprehensive Security Management
JMS 1.1 compliant	Application Server Neutral	Multiple transports (JMS, MQSeries)
Supports Multi platform and languages	Works with other MOM infrastructure	Monitoring and Document Tracking
Bridges to MQSeries and other MOM Infrastructure	Easily integrates with J2EE, .NET, COM	Distributed Deployment and Configuration Management
Easy upgrade to Tifosi	Easy upgrade to Tifosi Integrator	Dynamic Logging and Versioning

Solutions to satisfy your technical and financial challenges
Implement in stages, with incremental Return on Investment

Fiorano is #1 in Standards based integration

Meet us at
Booth No. 1502
at JavaOne

Fiorano
Enabling Change at the Speed of Thought™

Download Free Evaluation Software today at www.fiorano.com



J2ME



J2SE



J2EE



HOME

Jini/J2ME solution that overcomes the limitations of MIDP.

DineOut Corporation has a comprehensive database of restaurant listings, reviews, and ratings in different cities in the continental United States. DineOut wants to build a system called RestaurantFinder that enables DineOut's subscribers to access these listings using their J2ME-capable cellphones. The software architecture for the RestaurantFinder system is shown in Figure 2.

The core components of the RestaurantFinder system are as follows.

RestaurantFinder Jini Service

The RestaurantFinder Jini service provides the ability to

" A JINI SYSTEM USES THE NETWORK AS A FOUNDATION TO ENABLE SERVICE DISCOVERY AND SERVICE EXECUTION "

search for restaurant listings by city and cuisine. The definition for this service is given by the interface `rf.service`. RestaurantFinder (see Listing 1) and the Java proxy for this service is provided by the class `rf.service.RestaurantFinderImpl` (see Listing 2). (Listings 2-5 can be downloaded from www.sys-con.com/java/sourcec.cfm.) Using Jini to develop the RestaurantFinder service has the following benefits:

1. Provides a fault tolerant infrastructure by hosting multiple instances of the RestaurantFinder service on different nodes in the network. Thus, if there's a failure of one node, users will not lose access to the service.
2. Enables an administrator to add, remove, and move instances of the RestaurantFinder service from the network without affecting the overall operation of the system.

Jini Lookup Service

The lookup service manages a persistent store of service registrations. The J2ME client uses the Jini lookup service to find and locate the RestaurantFinder Jini service. Using the lookup service decouples the J2ME client from the actual service instance, enabling the Jini server to use multiple service instances to handle requests from the client.

J2ME Client

The MIDP client shown in Figure 3 will access the RestaurantFinder Jini service.

A Jini client developed using the J2ME technology has the following advantages over a traditional thin-client WAP-

enabled cellphone:

1. It provides a rich user interface to DineOut's subscribers by supporting features such as on-device validation (as opposed to server-side validation) and persistence to save restaurant listings. Thus, users get a more responsive and easier-to-use solution.
2. The client can operate in connected or disconnected mode. Although users won't have access to all capabilities when operating in disconnected mode, they can at least browse through saved listings, and could potentially also perform searches if restaurant data were persisted to the device.

J2ME Client Proxy

A MIDP device runs on the Connected Limited Device Configuration (CLDC). The CLDC doesn't allow the user to load classes at runtime. This prevents the MIDP client from acting like a regular Jini client, which has the ability to download and execute the service proxy classes. To work around this restriction, the RestaurantFinder service uses a Java servlet given by class `rf.servlet.ControllerServlet` (see Listing 3) as a Jini proxy. This proxy communicates with the RestaurantFinder Jini service on behalf of the MIDP client. The communication between the servlet and the MIDP client takes place over HTTP as shown in Figure 2.

Developing the RestaurantFinder System

Now that we're familiar with RestaurantFinder's architecture, let's have a closer look at what's involved in developing the RestaurantFinder system.

Developing the Jini Service

The first step in developing the RestaurantFinder service is defining the service interface `rf.service.RestaurantFinder` (see Listing 1). This interface defines the method `getRestaurants`, which takes a city name and cuisine as an argument, and returns an array of `rf.service.Restaurant` objects. This interface defines the contract between the Jini client and the Jini service.

Next, we develop the Java proxy class `rf.service.RestaurantFinderImpl` (see Listing 2) that implements the RestaurantFinder interface. During service registration, this Java proxy is uploaded from the RestaurantFinder service (see Listing 4) to the lookup service as shown in Figure 4. The proxy shields the client from the actual communication with the Jini service. When a client looks up a service, the lookup

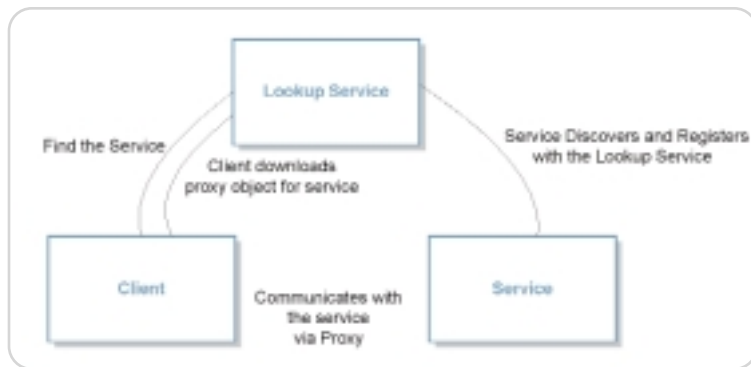


Figure 1 Jini architecture

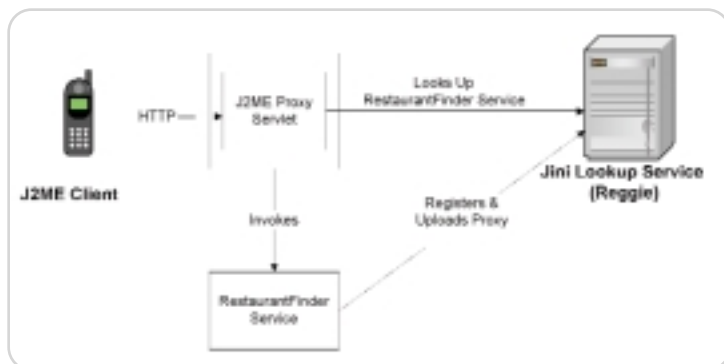


Figure 2 RestaurantFinder software architecture

The background of the top half of the page is a composite image. It features a globe in the center, with binary code (0s and 1s) flowing around it in various directions. The color palette transitions from blue on the left to orange and yellow on the right. The text 'Information in Action' is overlaid on the globe.

Information in Action

Everything works better
when everything works together.™

SYBASE
TechWave 2003
USER TRAINING & SOLUTIONS CONFERENCE

August 4-8, 2003

Gaylord Palms™ Resort & Convention Center
Orlando, Florida USA

www.sybase.com/techwave2003



service returns the proxy object for the service. Any classes required by the proxy are dynamically loaded over the network. In our example, the RestaurantFinder service is executed locally as part of the Jini client's (J2ME Client Proxy in our case) virtual machine. However, the Java proxy could easily be changed to communicate using RMI with a remote RestaurantFinder service.

Finally, the RestaurantFinder service is registered with the Jini lookup service (Reggie). The class ServiceRegistrationClient (see Listing 4) performs this operation.

Consider the following lines of code

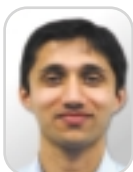
```
ldm = new LookupDiscoveryManager(new String[]{"iguanas"}, null, null);
jm = new JoinManager(new rf.service.RestaurantFinderImpl(),
    null,
    new ServiceRegistrationListener(),
    ldm,
    null);
```

The first line creates an instance of LookupDiscoveryManager to locate the group "iguanas". This is a utility class provided by the Jini reference implementation for discovering groups that a service is interested in joining. Groups provide a partition mechanism for the physical Jini network. A Jini service can belong to one or more groups. A Jini client has to join a group before accessing the services of the group. The RestaurantFinder service will join the "iguanas" group. This group is created and serviced by Reggie. The names of the group(s) to be serviced by a Reggie are provided as command-line arguments while starting Reggie as shown below.



Ron Dearing is a senior software engineer at Cysive, Inc., based in Reston, VA. He has worked in the software industry for over six years, and is currently working in the product development group on the Cymbio Interaction Server. Ron has a BS in computer science from North Carolina A&T.

rdearing@cysive.com



Nikhil Patil has worked in the software industry for over six years, and is a product manager at Cysive, Inc., where he works on developing the strategic vision for the Cymbio Interaction Server. A Sun Certified Java Developer, Nikhil has an MS in industrial engineering and a BE in mechanical engineering.

npatil@cysive.com

The next line creates an instance of JoinManager. This is another utility class that provides a convenient way to register a Jini service with Reggie (lookup service). In our example, an instance of the JoinManager is instantiated with an instance of RestaurantFinderImpl proxy, rf.service.ServiceRegistrationListener, and LookupDiscoveryManager. The JoinManager will register the RestaurantFinder service and upload the Java proxy for the server to the lookup service.

Developing the J2ME Client Proxy

Listing 3 shows the source code for the servlet rf.servlet.ControllerServlet that acts as a proxy for the J2ME client. Upon initialization, this servlet locates the RestaurantFinder service using the lookup service and dynamically loads the Java proxy from the Jini lookup service (see Figure 4).

The doPost method of the servlet decodes the request from a J2ME client, invokes the RestaurantFinder service,

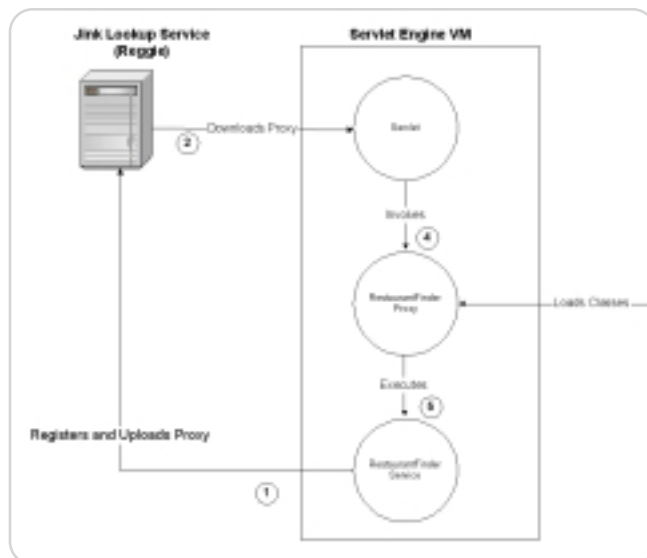


Figure 4 Dynamic class loading in a Jini network

and returns the results back to the J2ME client. Note that the ControllerServlet only accesses the Jini service through the interface rf.service.RestaurantFinder, thus decoupling the servlet from the implementation of the Jini service.

Developing the J2ME Client

The last piece of the puzzle is the J2ME client, which is developed as a MIDlet (see Listing 5) called rf.midlet.RestaurantFinderMIDlet. MIDlets are classes that extend the

"USING A SERVLET AS A PROXY BETWEEN THE J2ME CLIENT AND THE JINI SERVICE IS A STEP IN THE RIGHT DIRECTION FOR INTEGRATING J2ME CLIENTS IN A JINI NETWORK"

class javax.microedition.midlet.MIDlet and run on J2ME devices that support the MIDP profile. The execution and the life cycle of a MIDlet are controlled by special software on J2ME devices, called Application Management Software. MIDlets are developed on regular desktops and then deployed to smaller devices. The example in this article was developed using the MIDP reference implementation from Sun.

Execution of the RestaurantFinderMIDlet starts with the invocation of the startAPP method by the Application Management Software. This method initializes the J2ME client and sets up a search form as the current display for the application (see Figure 3). Using this form the user can select the name of the city and the type of cuisine for his or her restaurant search criteria. After selecting the criteria, the user can click on the "find" button to perform a search for restaurants.

Clicking the "find" button triggers the command findCommand. In J2ME, a command is something a user can use to interact with the client, e.g., clicking on the "find" button. In response to the command, the MIDlet opens up a connection to the J2ME Proxy Servlet and performs an HTTP GET as shown in method getResult. The J2ME Proxy



Figure 3 J2ME client



Figure 5 Search results



The RackSaver BladeRack XT, powered by the AMD Opteron™ processor, enables you to easily transition to 64-bit technology at your desired pace with increased 32-bit performance.

- The First 32-bit and 64-bit Supercomputing Cluster - AVAILABLE NOW!
- Up to 44 Nodes and 88 AMD Opteron™ Processors
- Over 700GB of DDR Memory Increases Performance and Scalability
- Configure the NEW BladeRack XT Online at www.racksaver.com

RackSaver® has teamed up with AMD to bring you the most powerful 64-bit super-computing clusters available on the market today. Increasing density in your server room is now even easier. More powerful, high-density clusters allow you to improve productivity and budget efficiency.



RackSaver servers utilize the AMD Opteron processor to enable customers to build high-density clusters using standard, cost-effective components instead of proprietary, high priced, RISC-based server components. Discover why industry leaders rely on RackSaver.

Check Out RackSaver's Other AMD Opteron-based Products



The RS-1164/op
1U Dual AMD Opteron™ Server

- 1U Dual Server design harnesses the power of the AMD Opteron™ processor to deliver new levels of performance and manageability into a dense 1U form factor.
- Server allows the power of 64-bit performance without sacrificing 32-bit performance.
- Enterprise-class, integrated systems and power management.



The QuatreX-64
The Highest-Performing
x86 4P Server, EVER!

- Standard 4U form-factor, tool-less access, and full extension rails.
- Massive storage capabilities with four hot-swap SCSI drives and slim-line CD-ROM and floppy options.
- Learn more about the QuatreX-64 online at www.racksaver.com.



Configure Your Server Today Using SuSE

RackSaver proudly supports SuSE Linux Enterprise Server 8 on all of its current servers and clusters. By using SuSE Linux Enterprise Server, you obtain a comprehensive no-worry package: including security, scalability, and around the clock availability. Furthermore, SuSE Linux Enterprise Server is scalable within a hardware architecture as well as across hardware boundaries.



RackSaver®

Leader in High-Density Servers!

Configure the all new QuatreX-64, BladeRack XT, and other AMD Opteron-based servers online at

<http://opteron.racksaver.com>

or call us toll-free at (888) 942-3800



Servlet (rf.servlet.ControllerServlet) returns a list of restaurants in the form of an encoded string. We've used an encoded string to keep the application simple; however, in real systems XML can be used for exchanging information between the J2ME client and the servlet.

On receiving the results of the search query, the list of restaurants is displayed to the user (see Figure 5).

The Road Ahead

Using a servlet as a proxy between the J2ME client and the Jini service is a step in the right direction for integrating J2ME clients in a Jini network. However, to fully enable J2ME clients as "true" Jini citizens, we can use the surrogate architecture (<http://surrogate.jini.org>). This architecture enables the J2ME clients to be part of the Jini network using a surrogate executing within a surrogate host. The surrogate knows of the device's capabilities and operates within the Jini federation. Services in the federation may use the surrogate just as they would use any other service in the federation. The IP Interconnect Adapter specification identifies the requirements of how the surrogate communicates with the device. This surrogate architecture combined with J2ME enhancements, like the MIDP 2.0 push API, can lead to the development of rich wireless clients that can not only leverage the Jini infrastructure but can also be managed as Jini services. (At the time this article was written, the specification for the surrogate architecture and the Interconnect adapters was not yet fully defined.)

Conclusion

J2ME and Jini can be used to develop a highly portable, resilient service-oriented architecture that meets the needs of mobile users. By providing some simple workarounds to

current limitations with J2ME, it's possible to build these powerful solutions today. ☺

References

- <http://pandonia.canberra.edu.au/java/jini/tutorial/Jini.xml>
- Edwards, W. K. (2001). *Core Jini (2nd Edition)*. Prentice Hall PTR.
- Edwards, W. K., and Rodden, T. (2001). *Jini Example by Example*. Prentice Hall PTR.
- <http://java.sun.com/j2me/>
- Knudsen, J. (2003). *Wireless Java: Developing with J2ME*. APress.

Listing 1

```
package rf.services;

import java.rmi.RemoteException;

/**
 * Interface for the RestaurantFinder Jini Service. Jini clients using the
 * RestaurantFinder
 * service will be program against this interface.
 */
@author Nikhil Patil
public interface RestaurantFinder
{
    /**
     * Returns an array of Restaurants based on the name of the city and type of
     * cuisine.
     */
    public Restaurant[] getRestaurants(String city, String cuisine) throws
    RemoteException;
}
```



Build Incredible Interactive Diagrams

Feature Packed New JGo Release!

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

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


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

New Version 5.0!

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



Would you like instant Java code comprehension?



We take orders now!


Juliet by infotectonica.com

Simply **browse** through annotated code and comments - Juliet automatically shows all relevant information while you focus on reading.

Instantly **search** through huge codebases containing millions of lines of code using multiple strategies, including searches for code patterns.

Easily **answer** complex questions no other tool can handle.

Juliet is a registered trademark of infotectonica sa. Java is a trademark or registered trademark of Sun Microsystems Inc. in the US and other countries.



Real world **performance** management for live J2EE systems

Quest Central™ for J2EE

It shouldn't be happening, but it is – unexpected performance problems in your production J2EE application! Now's not the time for guesswork and random, piecemeal diagnostic tools – you need an integrated solution for managing live application performance. Accelerate detection, diagnosis and resolution of J2EE performance problems with Quest Central for J2EE.

Only Quest Central for J2EE provides production-ready application management with unparalleled diagnostic depth for every expert on the team, ensuring that your live J2EE applications continue to fire on all cylinders.

Foglight™	PerformaSure™	JProbe®
Detect critical J2EE problems 24x7 – automatically alert and trigger diagnostics	Diagnose problems across all tiers and components with unique transactional Tag and Follow technology	Resolve code-level performance with deep, award-winning diagnostics toolkit

See us at JavaOne!

Silver Sponsor, Booth #1501

Get the full story – attend an exclusive webinar:
<http://java.quest.com/qcj/jdj>





J2ME



J2SE



J2EE



HOME



P800

by Sony Ericsson

This review has, admittedly, been quite some time coming. Had I been looking at basic phone features, I could have produced something months ago – however, this magazine is not the mobile phone-geek's equivalent of the *Trainspotter's Almanac* (fortunately), and we have slightly more relevant details to discuss, such as exactly how well the P800 performs when running Java applications – so the review has taken considerably more time than I originally expected. But I'll get on to that in a

minute.
For the moment, and for those of you

who haven't already come across a review of Sony Ericsson's smart phone offering, I'll briefly run through what the P800 offers outside of the Java space (just in case you were hoping you'd picked up the *Phonespotter's Almanac*, after all).

From the pictures, you might have supposed the P800 is a large device, too hefty to fit in a pocket, something more like the briefcase-sized "mobiles" of the '80s. Nothing could be further from the truth: it weighs in at a mere 158 grams and in terms of dimensions (11.7x5.9x2.7 centimeters), fits quite comfortably in the palm of your hand, and it's not so small that you feel as if your hand is wrapped around it twice (unless of course you have hands like a proverbial Goliath).

The P800 is triband GSM (900, 1800, and 1900 MHz), so you can use it almost anywhere and includes infrared and Bluetooth connectivity, just to make it that extra bit easier to drain those batteries on the go. The touchscreen is 208x320 pixels, 12-bit color, and with the flip open, seems to me just as functional in size as any other

PDA on the market: neither too small nor too large. The software offerings bundled on the phone are the usual PDA-like features: Contacts, Calendar (scheduling), Task list, Jotter (for notes), Voice memo, Calculator, Clock, Viewer (a text reader), plus a few

games and miscellaneous items like the picture viewer (for photos taken with the P800's built-in camera). The messaging application is impressive, integrating

SMS, MMS, and e-mail

Sony Ericsson

Sony Ericsson House
202 Hammersmith Road
London W67DN

Phone: +44(0)208 762 58 00

Fax: +44(0)208 762 58 87

Web: www.sonyericsson.com

into the one system, and the Internet browser includes HTML and cHTML support, as well as WAP/WML.

There are a couple of obvious omissions in that list: there is no mini-spreadsheet equivalent, nor is there a cut-down word processing program. The lack of a word processor isn't a major issue for me – for the moment – as I'm still trying to locate an external keyboard that will work with the phone. Despite the fact that the P800's handwriting recognition is, quite frankly, classy, I have no urge to spend a huge amount of time writing without a keyboard. However, I have occasionally missed having something more advanced than the simple calculator bundled on the device.

Java support is where matters get a whole lot more complicated.

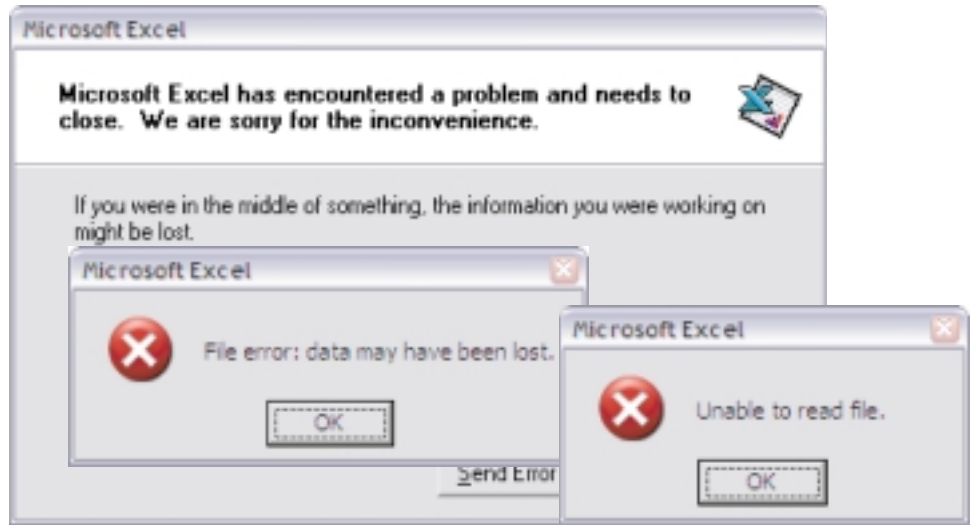
The P800 supports Java in two modes: J2ME (CLDC/MIDP1 support) and PersonalJava. The J2ME side of things is easier to get started, so I'll talk about this first. You can install MIDlets either via the USB docking station, or downloaded over the Internet using the built-in browser. Downloading is a simple process of entering the URL to the JAR file in the browser and waiting for it to transfer, and then selecting "Install" when prompted. Note that I said JAR there, and not JAD, since I was unable to get the P800 to install from an application descriptor, no matter how I tilted and shook the phone.

MIDlet performance is impressive. There were no noticeable hiccups in simple animations (even where there was a noticeable GC hiccup in the emulated MIDlet running on my laptop), and the phone seemed able to maintain a perfectly acceptable 20fps when drawing 40 primitives simultaneously to the screen.

My one complaint regarding the P800's MIDP support – and I've mentioned this before, elsewhere – is that a MIDlet will only run with the phone's flip keypad open. The phone operates in two distinct ways: with the flip keypad closed, it works like a normal



is it really free...



if it costs you a customer?

Other Java™ Excel™ tools claim... they "have the best support reading and writing Excel™ workbooks currently available." So we modified and wrote some of our customer's Excel™ files using both our competitor's tools and ExtenXLS with identical code.*

The results? While ExtenXLS reliably generated error-free Excel™ files, our competitor's Excel™ output was either corrupted or opened with errors – and that was when it didn't crash on reading the file!

Best-in-class Excel™ compatibility, robust features, and a company dedicated to its products.

That's why companies such as HP, Ford, JP Morgan, GE, and Disney trust ExtenXLS – and you can too.

ExtenXLS – the better Java™- Excel™ API.

Java is a Registered Trademark of Sun Microsystems Inc. Excel is a Registered Trademark of Microsoft Corporation. Jakarta POI is a Registered Trademark of the Apache Software Foundation.
* Comparison based on ExtenXLS.jar version 2.2 dated 5/15/03 and Apache POI version 1.10 (jakarta-poi-1.10.0-dev-20030222.jar). Copyright 2003 Extentech Inc.

JDJ Sale: Get \$200 off ExtenXLS Java™ XLS Toolkit through June 2003**

Use this special link to get your discount: <http://www.extentech.com/jdjsale/>

** \$200 off ExtenXLS Java XLS Toolkit: full version and ESD version only. Not valid with other offers.

exten**XLS**² version

- ✓ Fully supported
- ✓ Commercial-grade Excel™ compatibility
- ✓ Move and re-order WorkSheets
- ✓ Protect and unprotect WorkSheets
- ✓ Best-in-class 3D Named Range support
- ✓ Add hyperlinks to Cells for drill-down reports
- ✓ Serialize and copy Sheets between WorkBooks
- ✓ Total control of fonts and cell formatting
- ✓ Comprehensive sample code and templates
- ✓ Streamlined, easy to use API
- ✓ Add Java™ Dates directly to WorkSheets
- ✓ Used worldwide by global 1000 companies



phone; with the flip open, it operates in PDA mode. Controlling a MIDlet (especially games) using the touch-screen buttons is woefully difficult – either a process of fingers and thumbs smearing the screen, or trying to touch buttons rapidly with the pen. If a MIDlet could run in closed/normal phone mode (which seems to me the more logical place for a MIDlet to appear), the user experience would be considerably improved. This is a design decision in which I completely fail to see the logic.

The P800's PersonalJava support is the reason why this review has taken so long to deliver. Rather than just deploying a JAR file onto the phone in order to run a PJava application, your JAR must be packaged into a .SIS installation file before it will install. This would be fine if creating a SIS file was a simple process; however, it's quite patently not. To start with, the (Windows-only) SDK is a bloated (and I mean bloated) 270MB+ installation, in a set of 5x51MB and 1x16MB WinZip 8 files. If you're like me and spend most of your time trapped behind a pre-historic, nonbroadband hookup, you'll view the size of that install with something akin to horror. Installation is not straightforward either: it took me about three or four attempts to get the SDK actually installed, with numerous reboots in between.

On the subject of that Windows-only support, this is one of the more bizarre features of the SDK: large chunks of it are actually Java. And parts of it are Perl...and parts of it are native code. All packaged up in an almost-incomprehensible directory structure. To someone who has been used to the simplicity of the Java SDK, this is culture shock to the extreme (and I realize just how lucky we have actually been with Sun's efforts). In fact, had I personally been the perpetrator of this monstrosity, I would have, at the very least, fired myself, but more likely taken myself out back and administered a severe and justly deserved beating.

Had it not been for the efforts of a number of people on various forums who posted their experiences (and how to get around the problems), I would have given up, and pretended that I



hadn't noticed the P800 had PersonalJava support at all.

So, after much hair pulling, I can report that PersonalJava performance is, on the whole, average. In terms of animation, you might see no more than four to six frames per second when targeting the full-screen size (but perhaps slightly higher, if you rely on the standard graphics primitives). This sluggish graphical performance might not bother those of you who are developing business-oriented applications, but it very much depends upon the type of application. Experience in the past suggests that poor performance in the graphics department usually indicates a less snappy performance overall in any GUI app.

Of course, this is not entirely unex-

pected. PersonalJava is, by now, old technology. I would guess that many of the Hotspot-type performance improvements you would hope to see in the latest VM supporting the Personal Profile won't have been included in the P800's PersonalJava Virtual Machine. I'm crossing my fingers that newer versions of the Symbian UIQ operating system (the OS powering the P800) will support newer VMs, which is when you might hope to see a lot more developers interested in writing larger-scale Java applications for compatible devices.

To sum up my experiences with the P800, it has been a mixture of both good and bad. The good is the MIDP support, the clean, easy-to-use user interface, and the style and feel of the phone in general. The bad is the SDK and the lack of Linux support, both for connecting to the phone via USB and for developing more advanced Java applications than the simpler MIDlets that you have to package into a .SIS.

Assuming the Linux support changes in the future, and an upgrade comes along that updates the phone's Java support to MIDP 2 and the Personal Profile, the P800 would then become my perfect Java device. ☺

References

- *A good, detailed review of the P800:* www.infosyncworld.com/reviews/n/3067.html
- *If you're stuck with the SDK, this is a good place to look:* www.my-symbian.com/forum/

Snapshot

Pros

- Excellent MIDP 1.0 support. There is a possibility of MIDP 2 with a later version of the Symbian UIQ OS.
- A comprehensive set of features, both software, hardware, and especially connectivity.

Cons

- Lack of Linux support both for connecting to the phone and developing applications using the SDK.
- A bloated, virtually crippled SDK with incomplete and sometimes incorrect documentation.
- PersonalJava performance is just average. Acceptable for business applications, but if you were hoping to write multimedia games for the device, you'll be struggling.

The most powerful tool in information security does not come out of a box.



It comes from within.

Energize your brain, expand your knowledge base. Learn from the experts who are doing groundbreaking work. Network with delegates from all corners of the globe at the Black Hat Briefings USA 2003—the world's premier technical security event.



Black Hat®

Briefings & Training USA

July 28-31, 2003 • Caesars Palace, Las Vegas
Briefings: 2 days, 40 speakers • Training: 2 days, 8 topics

www.blackhat.com

for updates and to register or call +1.916.853.8555

diamond sponsor



platinum sponsors



gold sponsors



silver sponsor





Mike Fichtelman

Sex Machine Bean

A Java neural network built on ABLE

There's a carefully understated saying, attributed to the ancient Chinese: "May you live in interesting times." While at first glance living in interesting times might have been construed as a blessing, we who live in the present times know that interesting can be a curse. We've taken complexity to a new order of magnitude. It's hard to know what, exactly, we should be interested in. In these interesting times of information overload, it's often difficult to separate the wheat from the chaff.

Those of us in the business of computers know something that isn't always obvious to the layman, although a layman can often sense it. Our computers have not worked as advertised to reduce complexity in our lives, but in many ways have actually increased it. If your TV was as complicated to operate as your PC, you would have traded it in for a radio years ago. If you think about it for more than a couple of minutes, it's fairly obvious that this can't go on. We can't really afford to have every man, woman, and child on the planet become a computer technician. There's more to life than feeding the machines. If you feed a machine it will still be hungry tomorrow, but if you teach a machine how to feed itself, it can become autonomous. Autonomic computing has become a necessity.

GIGO

One of the reasons that computers haven't helped much in dealing with complexity is the simple fact that they do whatever they are programmed to do, and they are programmed by us. Anyone who's taken Computer Science 101 is familiar with the garbage in garbage out phenomenon. Although computers are much faster at certain kinds of serial processing tasks, like arithmetic, that can readily and clearly be expressed, the human brain is still far superior at handling complexity. While genes certainly play an important part in the programming of the human brain, it could be said that the human brain programs itself. In other words, it learns.

The modern French have another saying: "Vive la difference." This saying

assumes, naturally, that you can tell the difference well enough to enjoy it. In our interesting times, this isn't always easy. Perhaps it never was, for though the ancient Chinese had the concepts of yang and yin to denote the male and female principles underlying reality, the reality of the Tao as embodied in the famous symbol merges the two as they transition from one into the other. So it may not have been all that easy in the old days either.

If we often have trouble telling the difference, how can we expect a computer to do so? Computers are notorious for their poor tolerance of ambiguity. For example, we as humans don't have much trouble telling the difference between a male name and a female name, as long as the male or female name is one that we have learned as we've assimilated the other artifacts of our culture. Few humans from an English-speaking culture would deny that my wife's name, Eleanor, is female while my name, Michael, is male (I think there once was an actress named Michael, but I never could quite understand how – perhaps that was her intent).

A computer, on the other hand, would not "know" that Michael is a male name or Eleanor a female name unless the names were stored in a table of some kind. Indeed, that is precisely what we often do to "teach" a computer what it needs to "know." This is also precisely why computers are notoriously poor at tolerating ambiguity. If the foremost neurologists were to look inside my brain, or for that matter your brain, they would be hard pressed to locate where the table of cross references, from name to gender, is located. In fact, the very best of them could not locate it because it doesn't exist.

Electronic Brain

You don't have to be a brain surgeon to know that what does exist, in my brain and yours, are about two and a half pounds of tissue composed of neurons, axons, ganglia, and synapses. These countless cells combine in networks of networks of neurons of unimaginable (even when using the

imaginations that they make possible) depth and breadth. Nature has built with a couple of pounds of meat a massively parallel computer of unparalleled proportion, unrivaled even by acres of silicon and metal ones. It is only a slight exaggeration to say, then, that our table of name-gender associations is stored everywhere and at the same time nowhere in our brains.

There is a downside to this kind of approach, of course. Learning takes time. Unlike a computer, a database in my brain cannot simply be loaded with a set of name-gender associations. My brain must be repeatedly exposed to these associations, often over many years, and with many mistakes to firmly establish names that are male and names that are female. For a number of reasons, making mistakes is an important part of the learning process (Thomas Edison didn't think in terms of mistakes – he said he'd learned 5,000 ways how not to make a light bulb).

There is also a very bright side to this approach, which completely offsets the downside. Unlike computers, our brains are remarkably tolerant of ambiguity. Once the pattern of associations is established, our brains are exquisitely sensitive to the proximity of a partial pattern to the whole. You will not hesitate, for example, to recognize this name:

[Micha_l](#)

Most computer programs (those not employing the algorithms we're about to discuss) would not get a hit in the table of associations (even though 86% of the name is recognizable), and hence would fail to recognize it as a male name. The neural networks in our brains are quite resilient to noise, and can discern patterns even in information that has been heavily distorted.

Likewise, a neural network computer program can be used to recognize objects (classification) and trends (prediction) with considerable accuracy, as it uses a similar approach to pattern learning and recognition as that found in nature (see "Parallel Distributed Processing" in the reference section).



the Linux solution for all sizes

**Linux —
powering the growth
of your business.**



Call 1-888-hplinux
to speak with an HP Linux specialist
now, or visit www.hp.com/linux.

Linux solutions that reduce IT complexity,
increase agility, and deliver security all
backed by 24x7 HP support services.
Whatever your IT needs and size, HP has
a Linux solution customized to your needs.



Agent Building and Learning Environment

The Agent Building and Learning Environment, or ABLE, is a complete environment for designing, testing, and implementing Java-based artificial intelligence agents. The ABLE framework was developed by researchers at IBM's T.J. Watson Research Center. In terms of artificial intelligence features supported, ABLE covers the waterfront, and you would be hard pressed to think of an artificial intelligence paradigm that it does not employ. It is an AI tour de force.

Each paradigm is implemented as a JavaBean, and the collected beans are called AbleBeans. While this article focuses only on a single paradigm, that of neural network classification, there are many others, including Temporal Difference Learning, Naïve Bayes, and Radial Basis Function.

In addition to these artificial intelligence paradigms, ABLE provides agent beans for buyer/seller conversation logic, and for developing agents to automatically tune (autotune) computer systems and networks. There's also a complete ABLE Ruleset Language (ARL) for incorporating expert system paradigms, such as forward and backward chaining and predicate and fuzzy logic, into an application.

In the spirit of the best IDEs, ABLE enables you to incorporate intelligent agents into your applications without necessarily being an expert in AI. ABLE allows you to focus on what you want to do and not on how it is done.

The first step is to download ABLE from the IBM alphaWorks Web site (www.alphaWorks.ibm.com/tech/able). The Web site not only contains the distributions for Windows and Linux, but it has a lot of information about ABLE, including a moderated news group.

There are currently five downloads available, but the two related to Linux are in tar/gzip format. One tar file contains the executable and the other contains the help and Javadoc files. Even if space is an issue, I would recommend installing both, since the help files contain a lot of valuable information related to installation and use, as well as a complete tutorial. One important consideration (but apparently often overlooked – and not just by me, if the newsgroup threads are an indicator) is that you must be sure to have Java2 v1.3 installed on your system (and this is true whether you use Linux or Windows).

Simply untar the compressed file to a target directory (for example, able) and the ABLE files and documentation will be stored appropriately in a directory tree below it. This will also create a couple of preferences files. Next, change directories to go to the `..\able1_4_0\bin` directory. You will find several shell scripts there. You can execute either `runnit.sh` or `runjas.sh` (ABLE Java Agent Services) to start the ABLE Editor. These scripts make the startup of the ABLE Editor very simple, since all the classpath and JAR information is already provided. Before you start either of these scripts, however, you must ensure that you've either already set a `JAVA_HOME` environment variable to the location of Java2 v1.3, or you will need to set it within the script by editing it. If all goes as it should, you will see the ABLE Editor displayed before you.

The ABLE Editor is a full featured IDE (written in Java) that can be used to design, test, and debug intelligent agents. It's extremely easy to use, and it takes much of the work out of this process. Before going much further, however, it would probably be a good idea to validate that everything is fully functional at this point. A simple way to do that is to open and run the Animal neural network classification example provided with ABLE. This simple example completely exercises the features of ABLE and is based on the old familiar Prolog animal classification problem. For those who don't remember, the problem is to classify various animals (panther, zebra, etc.) based on their various characteristics (four legs, stripes, etc.). From the dropdown menu, select File, then Open Agent, then navigate to the neural directory and select `animal.ser`. This will load the Animal NeuralClassifierAgent bean, which is comprised of Import, TestImport, InFilter, OutFilter, and BackPropagation components. There are also Inspectors associated with the Infilter, Outfilter, and BackPropagation components. I'll explain each of these in a moment, but for now you can simply validate that they have loaded and displayed properly. If they have, you can hit the Run button (circle of arrows) on the ABLE Editor top panel. This will start the training or testing of the neural network. If you cannot get this far, you'll need to do a bit of research to identify the problem.

Based on my experience, I can tell you that the likelihood of an ABLE

problem is small at this point. More likely there's either a Java or a Linux problem. Problems related to Java might include version, location, and font properties. Linux problems could be similar, and include things like resolution settings and file locations. If you run into a situation where ABLE won't start or the components won't load or execute properly, it's likely something in your environment needs to be configured appropriately. A good resource is the newsgroup mentioned earlier, at the ABLE alphaWorks site. Once validation is complete and you're confident that ABLE is working as it should, you can proceed to design your own intelligent agents.

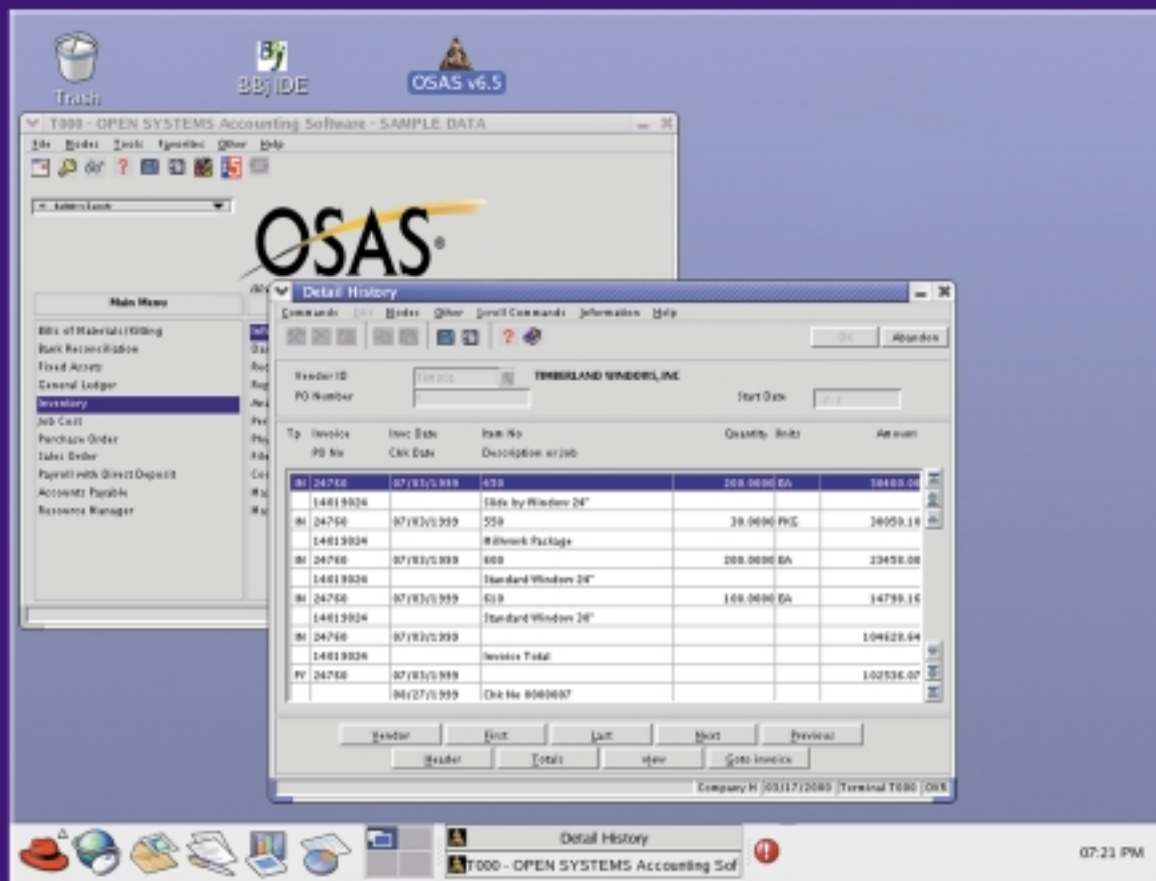
Sex, Machines, and JavaBeans

The SexMachineBean agent I'm about to describe is based on the neural network classification paradigm. (The source code for this article can be downloaded from www.sys-con.com/java/sourcec.cfm.) As I said earlier, this agent can be used to discriminate between male and female names. By definition, a key element of neural network classification is the representation of the data. Since the neural network will be learning patterns in the data, it's extremely important to represent it appropriately. There are two parts to representing data to ABLE. The first is a file (which could be a DB) containing the data itself; the second is the data definition file. In the case of SexMachineBean, the data file contains records that look like this:

```
Avri elxxxxxxx female
Avri lxxxxxxx female
Ayde exxxxxxx female
Aydri axxxxxxx female
Azale exxxxxxx female
Azemin axxxxxxx female
Azi axxxxxxx female
Azile exxxxxxx female
Azucen axxxxxxx female
Azzolin axxxxxxx female
Aaronxxxxxxx male
Abaramxxxxxxx male
Abbe xxxxxxxx male
Abbot txxxxxxx male
Abdel xxxxxxxx male
Abdi xxxxxxxx male
Abdool xxxxxxxx male
Abdul xxxxxxxx male
Abdull axxxxxxx male
```

The data files, by convention, have suffixes of `.dat`. The data definition file (which has a suffix of `.dfn`) for this data looks like this:

For over 25 years, OPEN SYSTEMS® Accounting Software (OSAS®) has been one of the most widely used business management packages in the mid-market. It's also the ideal solution for Linux users.



- Powered by BASIS technology
- 17 accounting modules
- Many vertical applications
- Free source code for easy customization

Protect your investment with BASIS development tools and technology and join the thousands of companies that use OSAS to run their business operations.

Open Systems and BASIS...
Pioneering accounting solutions for Linux.

name1	categorical input
name2	categorical input
name3	categorical input
name4	categorical input
name5	categorical input
name6	categorical input
name7	categorical input
name8	categorical input
name9	categorical input
name10	categorical input
name11	categorical input
name12	categorical input
gender	categorical output

From the data definition file, you can see that there are 13 fields defined for each record or name/gender pair. This is simply because the longest name was 12 characters, and the gender field brings the total to 13. Since not every name occupied 12 characters, it was necessary to pad the names that didn't (I simply used the character *x*, but an underscore would have done just as well).

It's also important to create both training and test files. This can be as simple as creating two sets of data and definition files. In the case of `SexMachineBean`, the files are named `smb.dat` and `smb.dfn` for training, and `smbTest.dat` and `smbTest.dfn` for testing. These should be stored in the `examples/datafiles` directory. I'll explain how these are used in a moment, but first, go to the ABLE Editor and from the dropdown menu select `File`, then `New`, then `Default(com.ibm.able.agents.AbleDefaultAgent)`.

This, by the way, is the default upon opening the ABLE Editor; so if you've just opened the ABLE Editor, you don't need to perform these steps.

Next, go to the `Agents` tab and you'll see a number of icons. One of these looks like a vertical sorting bin, with two arrows pointing to separate slots in the bin. If you let your mouse hover over this icon, the descriptor `AbleNeuralClassifierAgent` will be displayed. Push this button, and a neural classifier agent will be added to the workspace. It will appear gray, since it has not yet been configured and therefore is disabled.

Then, highlight the `NeuralClassifierAgent` in the left pane of the workspace. To configure the new agent, simply right-click the mouse on the `NeuralClassifierAgent`. Select `Properties` from the list that will be displayed. You'll be presented with four tabs: `General`, `Neural Classifier`, `Connections`, and `Functions`. Select the `Neural Classifier` tab. The others can be ignored for the time being. For the `Training File Name`, click on the `Browse`

button, navigate to the `examples/datafiles` directory, and select `smb.dfn`. Do the same for the `Test File Name`, but select `smbTest.dfn`. Then, under `Hidden1`, enter a value of 10. This will create 10 hidden units on the first intermediate network layer. This can vary depending on the application, and may require some amount of experimentation for optimal training. This number of hidden units is adequate for training the network to recognize all female and male names beginning with the letter A. Finally, click the `Generate Beans` button, then click `OK`.

You'll notice that the ABLE Editor will have generated an `Import` bean, a `TestImport` bean, an `InFilter` bean, a `BackPropagation` bean, and an `OutFilter` bean. You'll also notice in the GUI panel on the right that the beans are appropriately connected as well. You can open the properties for each of these beans and review the information displayed to get a better understanding of the ABLE environment. Notice that both the `Import` and `TestImport` beans are connected to the `InFilter` bean, but that only one connection is enabled. This is because during the training of the neural network, the ABLE Editor will alternate between testing and training in order to validate what has been learned.

To view the progress of the training process, the ABLE Editor provides a facility for creating `Inspectors`. Creating an inspector for a bean is as simple as everything else in the ABLE Editor. To create `Inspectors` for the `InFilter` and `OutFilter` beans, highlight each one in turn, right-click, then select `Inspect`. Two new windows will open, and the values for each filter will be displayed in the corresponding window. The inspector for the `BackPropagation` bean is created the same way, except that in this case certain parameters should be selected – at least, I selected the following parameters that I found the most informative on training progress: `percentCorrect`, `percentIncorrect`, `percentUnknown`, `netEpoch`, and `netArchitecture`. Naturally, under optimal circumstances, `percentCorrect` should rise to 100, while `percentIncorrect` and `percentUnknown` fall to zero. If that doesn't happen, it will be a clear indication that you've either got a problem with data representation or with the neural network architecture you've defined. The amount of data is also a factor in training. For example, if you have a relatively small amount of data, but the neural network never trains or oscillates, you may want to

increase the number of hidden units used. Setting the network architecture is somewhat of an art, but there are many references available on the subject. The `netEpoch` parameter displays the number of passes of the training set, while the `netArchitecture` parameter displays the defined number of input, hidden, and output units. For any inspector, the parameters to be displayed are selected by choosing `Data`, then `Parameters` from the dropdown menu.

Learning About the Birds and the Bees

That's basically all there is to the configuration of `SexMachineBean`. To start training, simply highlight the `NeuralClassifierAgent` in the left pane of the workspace once more, then right-click the mouse on the `NeuralClassifierAgent` and select `Properties` from the list that will be displayed. Select the `Neural Classifier` tab. Press the `Start` button to begin training the neural network. Training will continue until one of three things happens: the minimum percent correct reaches the threshold you've set; the maximum number of passes reaches the threshold you've set; or you hit the `Stop` button. It's difficult to predict the training time required, since it will vary based on the data, the neural network architecture, and the speed of the machine. The relatively modest training set provided here should not take very long, so please be patient.

Once you have a trained neural network, you can save the neural network state in serialized form. This is true for all ABLE agents, not just for neural networks. To me, this is one of the most significant and valuable contributions of ABLE. ABLE allows you to simply store and reuse in your application whatever has been learned. After all, the ease of use provided by ABLE in designing and building a neural network would be somewhat academic if you couldn't take it any further. Since ABLE allows you to store a trained neural network in Java serialized form, these objects can be re-created anywhere that a JVM will run. Obviously, this provides the intelligent agents created with ABLE a high degree of mobility and versatility. To store an agent in serialized form, simply click `File` from the dropdown menu. You'll notice that below `File` are selections for `Save` and for `Export`. It is recommended that, when the agent is finished and set to be used in your application, the `Export` alternative should be used. At this point, just

– continued on page 92

Mike Fichtelman is a certified senior project manager at IBM supporting their Web hosting business. He has over 20 years' experience in the information technology field as a developer, designer, and project manager. Mike has an MBA from Hofstra University and his work has been published in a number of journals on subjects ranging from infrastructure to the development of wireless applications using Java, XML, and WAP. He also teaches an e-business course in the MBA program at the University of Phoenix

mikef@optonline.net

ASK YOUR MANAGER

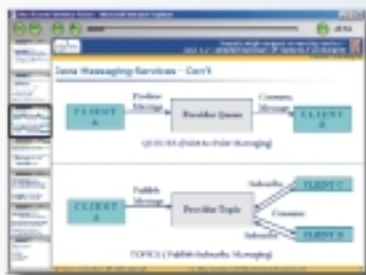
"Can we purchase the Isavix eLearning courses for our developers? They are short, high-impact, developed by industry experts, and most are **under \$50!**"

Self-service purchase, instant access ... visit <http://isavix.net> today!

FREE!

J2EE and .NET:
A Comparison

Introduction to
Web Services



How It Works!

The Isavix High-Impact eLearning Series™ is designed to deliver online training in the most efficient manner possible. Each course is approximately one to two hours in length with no "homework" assignments. The course slides/audio can be viewed/heard online on our web site in Macromedia Flash format. In addition, the course slides can be downloaded in PDF format for printing.

Partial List of Courses:

Introduction To Ant

This course will help you write your first build script, along with giving you an idea of what types of tasks Ant can do. This is a focused course, so you don't need to spend hours reading through books or online documentation to get to the heart of understanding Ant. You should go away from this course having a solid understanding of how Ant works, and how it can help you.

The Struts Framework

Struts is an open source framework useful in building web applications with Java Servlet and JavaServer Pages (JSP) technology. If you haven't used it yet, here's your chance to see why you should. This course focuses on how to use the Jakarta Struts project in your application development. Based on the Model-View-Controller pattern, the framework brings a significant advantage to your application development -- which explains the rapid growth in popularity of the Struts project.

Getting Started with XSLT

Over the last several years, XML has gained acceptance in a wide variety of business applications-but XML itself is not enough. XSLT, eXtensible Formatting Language Transformations, is one of the keys to bringing XML to life. It is a rich and versatile language that enables you to create HTML on the fly or transform business data from one XML "dialect" to another. This course will give you the essential knowledge you need to get started with XSLT.

Java Best Practices

Take your J2SE, J2EE, and J2ME skills to the next level with this informative, high-impact presentation of Java Best Practices from a noted author, mentor, and trainer.

Java has become a tremendously popular programming language in the last few years. As is the case with any new language, once the initial hype has died down and the excitement of the language's "newness" subsides, performance and efficiency become key.

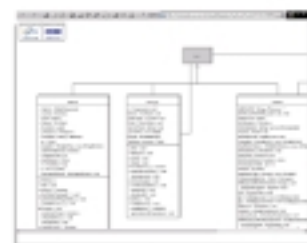
Java Performance Tuning

Developing with Java has many advantages. One of the key advantages is the capability to rapidly develop working applications. This ease of application development often comes at the expense of optimization. Developed applications are not always optimized for maximum speed and the most efficient use of memory. Over time many optimization techniques have been developed to optimize the performance of Java applications.

In this course you will learn many of the Java performance tuning techniques that are applied daily in the application development world. Once you become familiar with these techniques you will begin to use them instinctively as you write additional applications. Your reward will be applications that perform substantially more efficiently than un-optimized systems.

For more information, visit <http://isavix.net> or contact us today!

PLUS ... lots of FREE tools/resources for you!



My UML Diagrams

Class diagrams on-the-fly from your Java code!



Discussion Forums

Useful forums for J2EE and related technologies

Many more:

Code sharing, white papers, bookmarks, secure file storage, download links, and much more...

About Isavix

Isavix is an information-technology solutions company that provides custom services based on emerging technologies such as J2EE™, .NET™, Web Services, and many others. Since its inception in 1996, Isavix has built robust, secure, user-friendly, and cross-platform enterprise applications for many large and other leading organizations.

isavix
Delivering Emerging IT Solutions

isavix Corp., Herndon, VA, +1 703-468-3190, info@isavix.com



Ensemble Glider

by Ensemble Systems

Reviewed by
Ron Phillips

work with a lot of J2EE development tools. While some feel like solutions looking for a problem, every once in a while I run into one that feels like it was inspired by a developer's frustration at not being able to work quickly and effectively. Ensemble Glider from Ensemble Systems is that sort of tool.

What Is Glider?

Glider is an integrated development toolkit that accelerates J2EE development. It allows you to interactively create, build, and debug J2EE components before deploying to a server. Glider compresses the development cycle required to build and test EJBs and JSPs.

J2EE developers have to endure a fair amount of monkey motion in order to build and test J2EE components. Instead of having to endure a full cycle of code, compile, package, and deploy before debugging, Glider accelerates development by compressing that cycle into an interactive seamless coding and debugging session. It integrates with the Java compiler and debugger, and simulates a full-blown J2EE server, eliminating the need to package and deploy for testing.

Intermediate to advanced developers will get the most benefit from Glider. The tool doesn't provide extensive help for someone just learning about EJB development – you do need

to have a moderate amount of knowledge to be productive. It's fast and lightweight, and stays out of your way so you can get some real work done.

Getting Started

There are versions of Glider for the Eclipse shell, for Rational/IBM Rose and Rational/IBM XDE UML modelers, and a standalone version. My testing was done with the standalone version.

Installation is simple. It's distributed as a compressed file that you decompress into a target directory. Not having a minor installation program to create a launch menu was a minor inconvenience. Glider is started by launching the executable in the bin subdirectory. It automatically found my Java runtime and configured itself – I didn't have to do anything else for configuration.

If you are using the Java 1.4 runtime, make sure you have a recent 1.4.1 build. I encountered problems with earlier 1.4 JREs, but Ensemble's technical support cheerfully pointed me in the right direction.

Running Glider

I was pleasantly surprised at how quickly Glider starts up. Its user interface is simple and intuitive. I started by creating a Glider project – a workspace that holds EJB modules and Web applications. Once created, you can create new J2EE components, or add existing components.

The project workspace provides a project browser with tabs for viewing and navigating the project by its directory structure, package structure, EJB components, and Web components. This browser is the primary means of navigation.

Online help is available from the main menu bar. It's HTML based and, by default, directs your browser to the Ensemble Systems Web site. I downloaded the help files and configured the options to use them.

Ensemble Systems Inc.

Suite 280-5200 Hollybridge Way
Richmond, BC
V7C 4N3 Canada
Phone: 877.290.2662
Web: www.ensemble-systems.com

Specifications

Platforms: JDK 1.3.1 and JDK 1.4 (with Hot Swap support), Windows NT 4.x, Windows 2000 Professional SP2, Windows XP, Unix, Linux. Supports EJB 1.1 and EJB 2.0, JSP 1.2, Servlet 2.3 specifications
Pricing: \$199 introductory price until July 15 2003, \$499 after July 15.

Test Platform

2.1GHz desktop, 256MB RAM, 40GB disk, Windows XP SP1

Working with JSPs

You can create JSPs and associated Web components from the context menu of the Project Browser's Web tab. The new file is populated with a skeleton JSP. This skeleton is one of several templates provided with Glider. You can easily add your own templates or customize the stock templates provided.

Glider doesn't provide a lot of bells and whistles for JSP editing. I used the template mechanism to create a library of JSP tags, which would have been nice as supplied templates.

You can run your JSPs using the built-in server. Glider compiles the JSP, starts it up, and launches your Web browser.

Overall, I found Glider's editor a bit thin for HTML and JSP development, but Glider mindfully watches for changes made by an external source so you can use it with your favorite editor.

Working with EJBs

Like JSPs, EJBs are created from the context menu of the project browser. Glider generates the descriptor, implementation, home, and interface source for the bean. Glider has built-in tools to keep the interface, implementation, and bean descriptor in sync as you work. You can also flesh out the bean by editing the descriptor directly – Glider will update the source code from the descriptor.

Some other nice touches: the editor

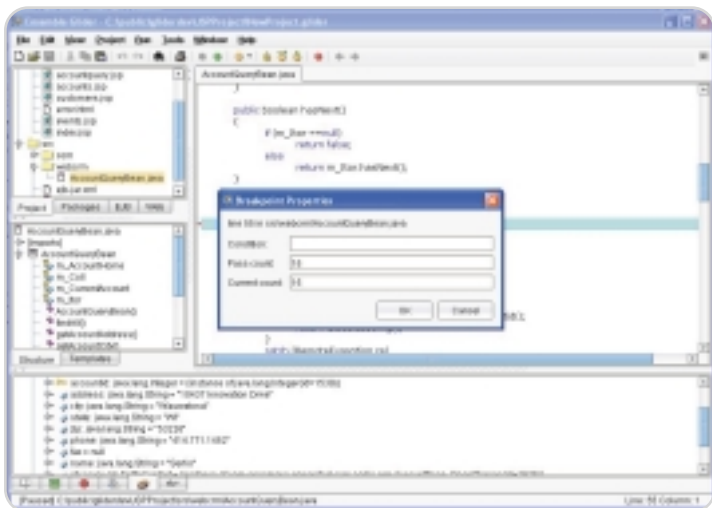


Figure 1 Glider's IDE's built-in debugger

– continued on page 86

Register before June 8, 2003, and Save \$100!

8th
Annual



JavaOneSM
Sun's 2003 Worldwide Java Developer Conference™

June 10–13, 2003

Pavilion | June 10–12, 2003

Moscone Center | San Francisco, CA

java.sun.com/javaone/sf

Innovation Everywhere

One Platform.
One Conference.

Be a part of the dynamic conference that brings the excellence of the Java™ platform to you. Come together with fellow developers from around the globe at the eighth annual JavaOneSM conference. Take advantage of the in-depth technical training that provides you with the innovative tools and solutions, while all new Hands-on Labs gives you the opportunity to test drive different technologies. This June, immerse yourself in the unparalleled JavaOne conference experience.

Sponsored by



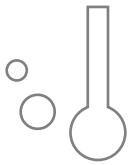
Produced by



Copyright © 2003 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Java, the Java Coffee Cup logo, JavaOne, the JavaOne logo, Java Developer Conference, Java Community Process, Java 2, EmbeddedJava, PersonalJava, 100% Pure Java, J2EE, J2ME, J2SE, Jini, Jiro, Solaris, Write Once, Run Anywhere, and all Java-based marks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

The JavaOneSM Conference — Everywhere Starts Here

Register before June 8, 2003, and save \$100 off the on-site price of \$1,995. Be sure to use Registration Code XXMG-333.



ObjectAssembler 2.5

Enterprise Edition by ObjectVenture, Inc.

Reviewed by
Adam Chace

Recently, I had the opportunity to work with the latest version of ObjectVenture's J2EE development tool: ObjectAssembler. ObjectVenture promotes ObjectAssembler 2.5 as a "smart" development tool that simplifies and accelerates J2EE development. This is a popular claim among Java tools, so I decided to put it to the test. My experience showed that ObjectAssembler lives up to the challenge.

Features

ObjectVenture offers two versions of their tool: Professional and Enterprise. I reviewed the full-featured Enterprise edition. ObjectAssembler Enterprise casts a broad net, offering the following features:

- Visual representation and management of J2EE components (EJBs,

JSPs, etc.) in the IDE, and real-time synchronization with code

- Visual Support for Struts, including full Tiles and Validator functionality
- Support for a language for defining and adhering to design patterns, known as Pattern Component Markup Language (PCML)
- Integration with JBuilder, Sun ONE Studio, and NetBeans
- Deployment to most J2EE application servers

Getting Started

ObjectAssembler Enterprise is available as a standalone environment or as a plug-in for Borland's JBuilder, Sun ONE Studio, or NetBeans. I downloaded the standalone evaluation version of ObjectAssembler 2.5.2, which is built atop the NetBeans IDE core. After downloading and completing the InstallAnywhere installation, I was up and running without incident in about three minutes. Just be sure you use a JDK that's 1.4 or higher.

One of the tool's most touted abilities is its visual "component" approach to J2EE development. To experience this, I created a very simple J2EE application containing a JSP, servlet, and session EJB that would reverse a string.

Component Workspace

ObjectAssembler's interface consists of three visual "workspaces." I spent most of my development time in ObjectAssembler's Component Workspace, where J2EE components can be visually created, modified, and maintained.

ObjectAssembler supports many components, which are created through a simple wizard. The components are broken up into several categories:

- **Struts:** Action, Plugin, ActionForm, DynaActionForm
- **Web:** JSP, lifecycle event, servlet, filter
- **EJB:** Entity bean, session bean, message-driven bean
- **General:** Interface, JavaBean, value object

I started by creating my servlet. Once I walked through the wizard, I

ObjectVenture, Inc.

Suite D208
2 Shaker Rd.
Shirley, MA 01464
Phone: 978.425.9654
Web: www.objectventure.com
E-mail: info@objectventure.com

Specifications

Platforms: Any platform with JDK 1.3.x support. Plugs into JBuilder, NetBeans, and Sun ONE studio or standalone
Pricing: Enterprise \$1,999; Professional \$499 (no pattern support)

noticed an example of ObjectAssembler's so-called "IntelliSynch" functionality that warned: "Servlet should implement a doPost or doGet" (see Figure 1). This warning seemed to go beyond enforcing interface adherence by offering general development tips. I found that the tool's tips and warnings do a good job (especially with novice developers) of finding silly errors. Beyond just identifying a problem, I was typically able to right-click on the warnings to implement a suggested fix with one-click.

While creating my JSP, the tool offered menus for adding directives, scriptlets, expressions, and tags. In addition to support for standard JSP tags, the tool allows importing of custom tag libraries and then enforces required attributes and tags. I couldn't help being impressed by how much easier JSP development could be with ObjectAssembler.

Last, I created a session bean using the EJB wizard (see Figure 2). In addition to wizards, the tool offers the ability to import and maintain existing components. This was reassuring to me, indicating that ObjectAssembler doesn't use proprietary repositories to maintain applications. To give IntelliSynch another test, I wrote the bean's method by hand, and, true to form, the tool visually added the method to the component in its Detail View.

Assembly Workspace

With the components for my simple application complete, I checked out the Assembly Workspace. The interface offered clean organization of my components, archives, and configuration files, with the ability to edit any item within the tool.

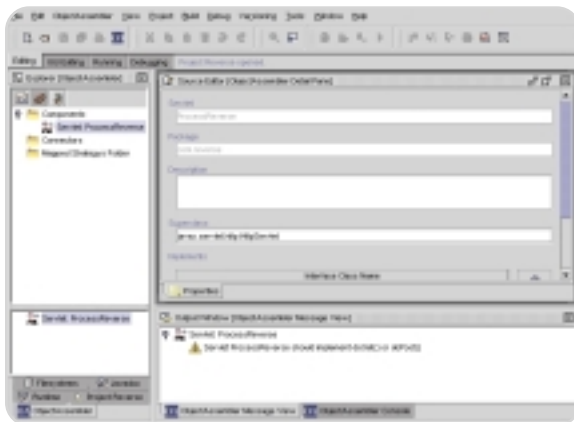


Figure 1 Component workspace

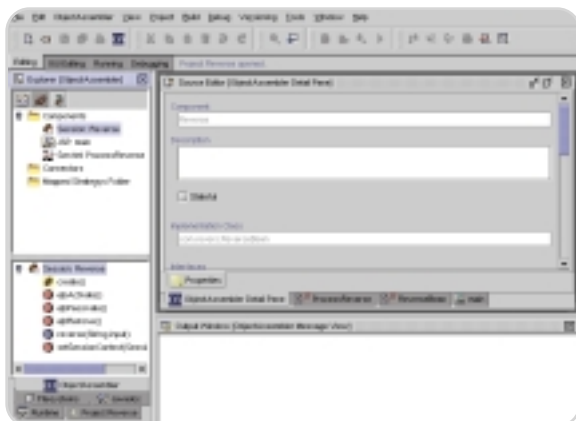


Figure 2 Editing an EJB



FROM ideas ▶ TO ▶ results

Rational User Conference 2003

August 24-28, 2003

Gaylord Palms Resort and Convention Center
Orlando, Florida

Join us for the Rational User Conference 2003, where we will help you make the leap from Ideas to Results! With 14 tracks for every member for the software development team, we'll provide all the technical insight and practical knowledge you need to leverage Rational's comprehensive set of development tools and services. Plus, we have an exciting line up of keynote speakers, Hands-on workshops, birds-of-a-feather sessions, and other networking opportunities.

Register Early and Save!

For more information or to register visit www.rational.com/ruc



To deploy, I right-clicked on the enterprise application and selected Generate EAR. ObjectAssembler compiled the necessary classes and packed up the EAR. The tool contains a “deployment versioning” feature that preserves application server-specific deployment descriptors from previous archives, and allows you to go back into the tool and make changes to your code without having to re-create the descriptors.

Patterns Workspace

A compelling feature of ObjectAssembler Enterprise is its ambitious support of software patterns, which allows you to capture designs and best practices for software development.

Traditionally, patterns have been described informally or as UML. ObjectVenture has developed an XML-based language to represent patterns. PCML, which ObjectVenture is working to standardize, allows patterns to be exchanged, modified, or extended in a human and machine-readable format. ObjectAssembler 2.5 Enterprise Edition ships with a version of Sun’s Java Center Patterns Catalog represented in PCML.

The impressive part of ObjectAssembler’s pattern support is that not

only can you easily create and modify patterns, but the tool actually tracks patterns after they are applied and can enforce them during development. If part of the pattern is not complete or the source code is modified and breaks the pattern, the tool informs the developer and assists in correcting the problem. I immediately thought of the potential; it’s like having an architect looking over the developer’s shoulder, keeping the project in sync with the original design.

Minor Issues

Overall, I had a very positive experience with ObjectAssembler. Like any tool, however, it’s not without problems. The documentation is not very comprehensive and I wanted more examples. ObjectAssembler’s user manual is adequate, but the online help was somewhat thin.

Another minor flaw is that ObjectAssembler’s integrated application server support is currently limited to WebLogic, JBoss, and Tomcat. ObjectAssembler does generate a J2EE-compliant archive that can be deployed to any J2EE-compliant application server. ObjectVenture indicated they would have additional application server integration available shortly.

Summary

In total, I was impressed with ObjectAssembler. I see its visual J2EE component and Struts development features as particularly useful for coaching development and reducing errors. The pattern support and the introduction of PCML show promise for enforcing design decisions throughout development, in particular on larger teams.

It’s not often that a development tool lives up to its claims of being helpful yet unobtrusive. ObjectVenture’s ObjectAssembler seems to meet these qualities and is well worth a look. ☺

Snapshot

Target Audience: J2EE developers

Level: Beginner to advanced

Pros:

- Fast, visual J2EE component building with lots of coaching, tips, and guidance
- Strong Struts support
- Ambitious support for creating, distributing, and enforcing software patterns

Cons:

- At present: a small user community
- So-so online documentation
- Limited built-in app server integration

Adam Chace is president of Chalk Creek Software, a J2EE consultancy in the U.S., and the coauthor of *JSP Tag Libraries* from Manning Press.

adam@chalkcreek.com

– continued from page 82

Ensemble Glider

provides syntax coloring for the Java code, and has pop-up code helpers for automatic code completion. The code templates automate commonly used coding constructs, and the visual editor for setting up container-managed relationships is simple and easy to use.

When you’re ready to start testing your EJB, Glider will generate a test client for you. From the test client code, simply add the code to instantiate your EJB from the bean home.

XDoclet Support

Glider supports XDoclet, which significantly simplifies EJBs. When you create your EJB, specify that you’ll be using XDoclet. All coding for an EJB is then done in the bean implementation file, and the interface and home are generated from XDoclet tags embedded in your Javadoc comments when you build. This eliminates redundant, error-prone work and keeps the implementation, home, and interface in sync.

Support for XDoclet in the editor is great – the context menu provides options to create the tags for you. You can easily create an EJB, add methods and

CMP features, or generate a test client and have it running in just a few minutes.

Debugging

Debugging is where Glider really shines – you can edit and debug incrementally without any deployment effort. Testing an EJB with a generated test client couldn’t be easier – just run the test client from within Glider. Glider will build the application and start the debugger. I was impressed with the rich debugging capabilities that Glider provides. You can create simple and conditional breakpoints, trace into JSP code, debug locally or remotely, inspect loaded EJBs with the integrated browser, and even save and preload the container’s loaded beans for testing specific scenarios (see Figure 1).

Summary

Glider simulates a J2EE server with remarkable completeness. For developing and debugging J2EE applications, it’s hard to beat the speed and convenience of this lightweight environment. The compressed code/test cycle encourages experimentation and creativity. The standalone version’s editor is basic but sufficient, and the environ-

ment works well in concert with external editors and other tools. If you are using Rational/IBM’s XDE or Rose, or eclipse.org’s Eclipse shell, you should check out those versions.

At a list price of \$499, Glider is a serious value for anyone doing J2EE development. At the introductory price of \$199, this is a no-brainer. See www.ensemble-systems.com/glider for more information. ☺

Snapshot

Target Audience: J2EE developers

Level: Intermediate to advanced

Pros:

- Fast and efficient tool for developing, debugging, and testing J2EE components (JSPs, servlets, EJBs)
- Full support for EJB2.0 including CMP, CMR, message-driven beans
- Good debugging tools
- XDoclet integration
- Works well with external editors and tools

Cons:

- Editor in standalone version could be more complete
- An installer could make installation and setup a little more convenient

Ron Phillips is vice president of software development at Serlio Software (www.serliosoft.com) in Milwaukee, WI. Serlio provides mentoring and consulting to large and mid-sized companies adopting software best practices into their development organizations. Ron has been designing and developing commercial software development tools for over 12 years and has coauthored several books and articles on Java technologies.

rkp@serliosoft.com

DARE TO COMPARE?



RIP OUT THIS FEATURE LIST AND SEE HOW YOUR SERVER STACKS UP AGAINST MACROMEDIA® JRUN™ 4 SERVER

NEW AND ENHANCED FEATURES OF JRUN 4:	JRUN 4	
Certified J2EE 1.3 Compatible	✓	
High performance web container with JSP 1.2 and servlet 2.3 support	✓	
Advanced EJB 2.0 support including local interfaces, message-driven beans, EJB-QL, and CMP 2.0	✓	
Additional Java® 2 APIs -- JMS 1.0.2, JTA 1.0, JAAS 1.0, JCA 1.0, JAXP 1.1, JavaMail 1.1, JDBC 2.0, JNDI 1.2, RMI/IIOP 1.0	✓	
Support for XML, SOAP, WSDL, and UDDI for web services	✓	
Built-in production quality HTTP server	✓	
Hot modification and auto deployment of applications	✓	
Integrated XDoclet support for simplified component deployment	✓	
Built-in Macromedia® Flash® MX Remoting services for high impact user interfaces	✓	
Type IV JDBC drivers included for Oracle®, MS SQL Server, IBM® DB2, Sybase®, and Informix®	✓	
Highly customizable and extensible JMX-based service-based architecture	✓	
JMX-enabled web-based management console	✓	
Advanced multi-tier server clustering enabled by Jini® technology.	✓	
Leading OS platforms, including Unix, Linux®, and Windows®	✓	
FULL-FEATURED PRODUCTION DEPLOYMENT	\$899/CPU	



**Java Compatible™
Enterprise Edition**

We've packed JRun 4 with features to make it faster, easier, or more affordable than any other J2EE server you can find. Don't just trust what you read, try it out yourself. You'll see why over 10,000 leading organizations deploy their J2EE applications on JRun.

Our **FREE** non-expiring developer version will get you started. Download JRun 4 today for **FREE** at: www.macromedia.com/go/jrun4jdj

Go ahead....we dare you.

Copyright 2002 Macromedia, Inc. All rights reserved. Macromedia, JRun, and the JRun logo are trademarks or registered trademarks of Macromedia, Inc., which may be registered in the United States and other countries. Jini, Java and Java-based trademarks and logos are the trademarks or registered trademarks of Sun Microsystems Inc., in the United States and other countries. IBM and Informix are registered trademarks of IBM Corporation. Sybase is a registered trademark of Sybase, Inc. Linux is a registered trademark of Linus Torvalds. Windows is a registered trademark of Microsoft Corporation. Oracle is a registered trademark of Oracle Corporation. All other trademarks are property of their respective owners.



**macromedia®
JRUN™ 4**

JDJ Editors' Choice Awards

The editors of *Java Developer's Journal* are in a unique position when it comes to Java development. All are active coders in their "day jobs," and they have the good fortune to getting a heads up on many of the latest and greatest software releases. They were asked to nominate three products from the last 12 months that they felt had not only made a major impact on their own development, but also on the Java community as a whole.

The following is a list of each editor's selections and the reason why they chose that product.

Alan Williamson

Editor-in-Chief

LISA from iTKO Corporation

LISA is a tool for developing and staging Web and XML-based regression and load tests.
www.itko.com/lisa

"This is a truly beautiful piece of software that fits snugly into anyone's testing environment. The ability to simply record and playback test results was the easiest thing I have seen in a long time."

Thinlets

Thinlets is a GUI toolkit, a single Java class that parses the hierarchy and properties of the GUI, handles user interaction, and calls business logic.

<http://sourceforge.net/projects/thinlet>

"This framework proves that there is life in AWT after all. If only AWT could have provided this from the start, then Flash may not have gotten the foothold on the browser that it did."

jEdit

jEdit is a cross-platform programmer's text editor written in Java, and developed by Slava Pestov and others. jEdit is released under the terms of the GNU General Public License.

www.jedit.org

"The plugin architecture alone wins my vote, even beating Eclipse's. To say this is just a text editor for code is definitely underselling this wonderful example of software engineering."

Joe Ottinger

J2EE Editor

IDEA from IntelliJ

IDEA is a Java IDE. Its development features includes industry setting refactoring support, and intelligent code editing assistance.

www.intellij.com/idea

"This is the editor that gets out of my way. It's actually changed how I write code, after being fairly set in my ways about code generation for over a decade. Going to any other editor is a chore for me now."

Orion Application Server

The Orion Application Server provides your application with a foundation that is robust, scalable, and easy to develop with.

www.orionserver.com

"Orion, like IDEA, does things the way I think they should be done: directly, with little fuss, and very quickly. Deployment in Orion is very simple, and the lack of required server-specific files means that I tend to develop on Orion and deploy on other servers...if I have to."

Ant from Apache

Apache Ant is a Java-based build tool. Ant is extended using Java classes.

<http://ant.apache.org/>

"Ant is the hammer of the Java world: without it, civilization might have progressed, but much more slowly than it has. Ant is one of the most useful build tools I have ever had the pleasure to use."

Jason Bell

J2SE Editor

Eclipse

The Eclipse Platform is designed for building integrated development environments (IDEs) that can be used to create applications as diverse as Web sites, embedded Java programs, C++ programs, and Enterprise JavaBeans.

www.eclipse.org/

"After being anti-IDE for so long I've finally caved in. It has nice CVS utils, project frameworks, code refactoring, and 'sensible' code generation (especially for beans). Add industry backing and a very fired up user base and you have one winning product."

Orion Application Server

The Orion Application Server provides your application with a foundation that is robust, scalable, and easy to develop with.

www.orionserver.com

"Any J2EE-compliant server where the installation is 'java -jar orion.jar -install' gets my vote any day. More fiddly than Tomcat but much faster."

Thinking In Java (3rd Edition)

by Bruce Eckel. Prentice Hall PTR, 2002.

"I've lost count how many times I've advised people to get hold of this book, whether by download or in print. I still suggest print as it very rarely leaves my side for those all important core Java tips."

Glen Cordrey

J2ME Editor

The Personal Profile, JSR-62

JSR-62 provides Java APIs for devices requiring a high degree of network connectivity such as applications for the home and office.

<http://wireless.java.sun.com/personal>

"I vote for Personal Profile, if only for the fact that it was finally completed. But I also think it's a pretty significant, and good, spec (for a first version)."

JBoss

JBoss is a Java application server developed in open source known for its ease of use, modularity, and simplicity.

www.jboss.org/index.html

"JBoss provides production quality for free, although it's still a bit rough around the edges."

Eclipse

The Eclipse Platform is designed for building integrated development environments (IDEs) that can be used to create applications as diverse as Web sites, embedded Java programs, C++ programs, and Enterprise JavaBeans.

www.eclipse.org

"Again, free goes a long way, especially with the support piling up behind it."

Jason Briggs

Contributing Editor

JAMiD (JAM interactive device)

Based on J2ME MIDP, the JAMiD accelerated Java platform will run MIDP 1.0/2.0 games and other multimedia MIDlets. Includes a built-in MP3 player.

www.jamid.com

"For bringing J2ME, specifically MIDP, to the GameBoy Advance - just plug the JAMiD cartridge into the GBA and you've got instant access to MIDP games on your handheld console."

Netx

Netx is an implementation of the Java Network Launching Protocol (JNLP). It downloads code over the network for applications and applets, caches it, and runs it in a secure environment.

<http://jnlp.sourceforge.net/netx>

"For bringing shared applications to the same VM - something Sun should have done a long time ago. Save memory by launching Web Start-style applications in the same virtual machine."

Tomcat from Apache

Tomcat is the servlet container that is used in the official Reference Implementation for the Java servlet and JavaServer Pages technologies.

<http://jakarta.apache.org/tomcat/index.html>

"For ever increasing performance - it's the one product that, every time I download a new version, manages to eke out a little more performance."

Linux Software Management is a Completely *Different* Animal

That's why Ximian® created **Red Carpet™ Enterprise™**,

the secure and centralized solution for enterprise software management you deploy completely behind your corporate firewall. It slashes your company's total cost of ownership by automating software updating and version control for Linux servers and desktops. Use it with leading Linux distributions like Red Hat, SuSE, Mandrake, Debian and more.

Red Carpet Enterprise will change the way you look at Linux.

Learn more. Get your free copy of "Linux Software Management 2003" at www.ximian.com/information/mgmt3



XIMIAN™

Enabling Enterprise Linux

Industry News

JBUILDER 9 Integrates All Phases of Java Application Life Cycle

(Scotts Valley, CA) – Borland Software Corporation has introduced Borland JBuilder 9, the latest version of the Java development solution, accelerating the development of Java-based applications for the enterprise. Borland also introduced Borland Optimizeit Suite 5.5, the latest version of the award-winning Borland performance assurance solution, engineered to help developers address performance issues early in the development life cycle.

www.borland.com

SeeBeyond Delivers First J2EE-Compatible Integration Platform

(Los Angeles) – SeeBeyond, a global provider of enterprise integration and composite application assembly solutions, has announced that it has successfully completed the certification process for Java 2 Platform, Enterprise Edition 1.3, compatibility with its SeeBeyond Integrated Composite Application Network (ICAN) Suite v5.0. In addition to providing a comprehensive J2EE-compatible integration platform, SeeBeyond is the first vendor to support the execution of its application integration functionality on third-party J2EE-compatible application servers.

www.seebeyond.com

Xora to Deliver Enterprise Application Data Access via the Sony Ericsson P800 Smartphone

(Mountain View, CA) – Sony Ericsson's P800 Smartphone, featuring PersonalJava and Java MIDP support, has been verified with Xora's J2ME solution to enable mobile access to Siebel, Clarify, SAP, PeopleSoft, Oracle, and other applications using Xora's

EnterpriseOne mobile platform. The P800 Smartphone and Xora EnterpriseOne deliver a turnkey solution to access any data, any time, any place.

www.xora.com

Q-Link Version 5.0 Brings J2EE Development to the Masses

(Los Angeles) – Q-Link Technologies has announced version 5.0 of their application development platform designed to reduce the overall complexity, time, and cost of developing J2EE applications. The Q-Link platform brings together three unique elements: a new application model, a suite of integrated Business Process Management (BPM) services, and a scalable architecture built on standards-based technologies.

www.qlinktech.com

Quest Releases PerformaSure 2.1

(Irvine, CA) – Quest Software, Inc., a provider of application management solutions, has released PerformaSure 2.1, introducing integration with Quest's Foglight application monitoring product. This release features dual-mode agent technology that enables a Foglight alert to automatically trigger deeper J2EE-centric diagnostics.

www.quest.com

Teamstudio Releases Analyzer for Java Edition 3

(Beverly, MA) – Teamstudio, Inc., a developer of agile software tools, has announced the release of Edition 3 of Teamstudio Analyzer for Java, its award-winning best-practices audit tool for Java developers. Edition 3 includes numerous feature enhancements beginning with autofix and reporting, in addition to 192 rules that address coding issues such as standards compliance, unused elements, common coding errors, and J2EE compliance.

www.teamstudio.com

Netegrity Extends Identity and Access Management Solution for WebSphere

(Waltham, MA) – Netegrity, Inc., a provider of identity and access management solutions, has expanded its support for IBM WebSphere-based applications. With this, Netegrity lever-

ages IBM's Trust Association Interceptor (TAI) to enable companies to incorporate their IBM WebSphere environments, including IBM WebSphere Application Server and IBM WebSphere Portal, into a unified and centrally managed security infrastructure.

www.netegrity.com

ReportingEngines Launches the Formula One Product Family

(Overland Park, Kansas) – ReportingEngines, a division of Actuate Corporation and provider of embedded reporting solutions for J2EE Web and application servers, has announced the immediate availability of the Formula One product line. The Formula One e.Report Engine and e.Spreadsheet Engine offer developers a full-featured, reporting toolset that can be embedded into any Java project or application deployed from a J2EE Web or application server.

The e.Report Engine extracts information from a variety of data sources including Java objects inside applications, databases, Enterprise JavaBeans and text files using public APIs. The e.Spreadsheet Engine is an API-driven component and report designer used to embed Excel-compatible spreadsheet reporting functionality into projects deployed from J2EE Web or application servers.

www.reportingengines.com

Mercury Interactive Acquires Performant

(Sunnyvale, CA) – Mercury Interactive Corporation, a provider of business technology optimization, has acquired Performant, Inc., a provider of comprehensive J2EE diagnostics software. Performant's technology pinpoints performance problems at the application code level, reducing the cost and time required to optimize J2EE applications. The Performant acquisition allows Mercury Interactive customers to diagnose J2EE performance issues across the application delivery and management cycle from testing through deployment to operations.

Under the terms of the merger agreement, Mercury Interactive paid \$22.5 million in cash.

www.mercuryinteractive.com

Sun Introduces the Java Card System Protection Profile

(Santa Clara, CA) – Sun Microsystems, Inc., has released the first Java Card System Protection Profile. This Protection Profile reduces the time and cost for Java Card licensees to complete security evaluations under Common Criteria. Sun will provide a reusable set of security requirements specifically for the Java Card platform.

Common Criteria was developed in 1996 as a combination of the Canadian Criteria, Federal Criteria, and Information Technology Security Evaluation Criteria (ITSEC) to provide standards for security evaluation of IT products that would be accepted in the international community.

www.sun.com

International Web Services Conference & Expo

WEST

Web Services Edge 2003



SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



**EXTENDING THE ENTERPRISE
WITH WEB SERVICES THROUGH JAVA,
.NET, WEBSHERE, MAC OS X
AND XML TECHNOLOGIES**




LINUX EDGE
conference & expo

web services EDGE
conference & expo

BOSTON
February 24-27, 2004

Featured technologies and topics will include:

- Focus on Java
- Focus on .NET
- Focus on WebSphere
- Focus on Mac OS X
- Focus on XML



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



Over 100 participating companies will display and demonstrate over 300 developer products and solutions.
Over 2,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.
Over 60 of the latest sessions on training, certifications, seminars, case studies, and panel discussions will deliver REAL World benefits, the industry pulse and proven strategies.

Contact information: U.S. Events: 201 802-3069 or e-mail grisha@sys-con.com



Java is a registered trademark of Sun Microsystems, .NET is a registered trademark of Microsoft, Mac OS X is a registered trademark of Apple Computer, Inc., WebSphere is a registered trademark of IBM. All other product names herein are the properties of their respective companies.

**WEB SERVICES EDGE WEST 2003
CALL FOR PAPERS NOW OPEN**
Submit your papers online at:
www.sys-con.com/webservices2003west



Letters to the Editor

JBoss – A J2EE-Compliant App Server?

I seriously question how Marc Fluery can be allowed to declare that JBoss is a J2EE-compliant application server (“Pulling at a Thread” by Alan Williamson [Vol. 8, issue 5]). Who has certified that it’s compliant? If JBoss is allowed to make this claim, what is to stop others from making it? Why should those who have paid for the certification have to be compared with those who haven’t? I say to Marc Fluery, JBoss should either ante up or drop the pretense that JBoss is a J2EE-compliant server and stop browbeating Sun for trying to maintain the J2EE label.



Kirk Pepperdine
kirk@javaperformancetuning.com

Another Analogy to Consider

It’s like Boeing and Airbus – at any one time one of them is in the lead but with so much money on the line they are always trying to beat each other (“Do Java and .NET Really Compete” by Joseph Ottinger [Vol. 8, issue 5]).

Java and .NET are the same deal as long as Sun/IBM and MS keep competing; the platforms will always be close in terms of functionality and ease of use, cost, etc. The true winners are the developer and business.

My advice – everyone should hedge their bets for maximum employability – first become a SQL/XML expert, then learn Java and C# (should be a relatively easy transition from either language).

Frank
kellyfj_2000@yahoo.com

There is a feature request for MDI. Go to <https://bugs.eclipse.org/bugs/> and search for PR number 29891. You can add yourself as a cc to this so you get notified of updates, and also add your comments and you can vote for the PR as well.

Joe Winchester

Some Valid Points but...

I completely agree with Nigel Thomas’s idea of a microkernel and pluggable modules for each “technology” (“Is J2EE Too Big for Its Own Good?” [Vol. 8, issue 4]). BEA and JBoss are already using JMX to do this, though at a very basic level. However, I don’t think that fragmenting a complete release is the right thing to do – it will become even



more of a mess. There are interdependencies between these modules (EJB, servlet, JNDI, etc.) that would be near

impossible to manage if there was no unifying process. And I don’t mean code, I mean *specifications*, which of course the code is based on.



Giriraj
giriraj.srivastava@elementn.com

Pratik Patel
prpatel@smarframeworks.com

What’s in a Name?

Although Ajit Sagar attempted to clarify the meanings of “proof-of-concept” and “prototype” in his editorial “The Proof Is in the Concept” (Vol. 8, issue 5), I have to disagree. Actually a proof-of-concept ought to do exactly that: give solid feedback on the basic characteristics of a specific concept. Therefore any statement with a POC has to prove it’s dependent on the concept and can’t stand on its own.



Michael Neumann
mn.de@shikou.org

MDI in SWT

I developed an application for my company in Eclipse SWT and I found it really fast and easy (“SWT: A Native Widget Toolkit for Java” by Steve Northover and Joe Winchester [Vol. 8, issue 4]). It’s outstanding Java development. The point where I got stuck was MDI things. That made me roll back for the time being to Swing. But I’m still actively looking for some solution to this need. Any comments?

Sex Machine Bean

– continued from page 80

choose a name (like smbA.ser) and proceed to store the bean.

To demonstrate how easy it is to include the trained neural network in an application and use it, I’ve prepared a simple Java application using WebSphere Studio Application Developer. Naturally, you can use whatever JRE engine you prefer. The application provides a panel in which a name can be entered. When you hit Enter/Return, the gender is returned. If you examine the code, you can see for yourself how straightforward this is. In a nutshell, the neural network objects

are restored, the name entered is parsed to conform to the neural network data representation, the parsed name is passed to the neural network as input, and the gender is passed back as output. As I mentioned earlier, if you enter a subset or incomplete name (for example, Adex instead of Adele), you’ll see that the neural network still maintains reasonable accuracy.

As you can see, the Agent Building and Learning Environment is as powerful as it is easy to learn and use. It puts the power of decades of artificial intelligence research within easy reach of any Java developer. In future articles, other facets of the ABLE framework will be

explored, such as its use to support autonomic computing applications. ☛

References

- Rumelhart, D., and McClelland, J. “Parallel Distributed Processing – Explorations in the Microstructure of Cognition.” A foundation text on neural networks. Vol. 1. Foundations. MIT Press/Bradford Books, 1986.
- IBM Systems Journal, v. 41 no. 3, 2002. Applications of Artificial Intelligence:
www.research.ibm.com/journal
- IBM Systems Journal, v. 42 no. 1, 2003. Autonomic Computing:
www.research.ibm.com/journal

THE SERVERSIDE SYMPOSIUM

BOSTON • JUNE 27-29 WEEKEND, 2003

"No Fluff, Just Stuff" Conference

You're Invited...

TheServerSide.com cordially invites you to come join us at **TheServerSide Symposium** – a special technical conference on **Enterprise Java (J2EE)** technologies. If you currently building Enterprise Java applications or plan to continue doing so in the next few years, this is not a show to be missed!

At this conference, you will learn about:

- Architecture best practices
- Design Patterns
- Open source frameworks and methodologies
- Web services

This limited-attendance conference is being held at the retreat-like Sheraton Colonial Hotel, near **Boston, Massachusetts**. We've chosen to hold the conference **June 27-29, 2003** - a long weekend, so that you don't have to take much time away from work. This means its easier to convince the boss!

The show is unique because it is the only show in the Java industry that promises you will learn an **unbelievable** amount of new, useful information about J2EE programming. We have an all-star lineup of speakers who are making a difference in the enterprise java development community, as well an extremely technical and advanced set of technical sessions of a nature that are not found at less specialized or more commercial shows.

Who Will Be There?

TheServerSide Symposium has assembled an incredible team of people who have been contributing to the enterprise java space and whose work has undoubtedly influenced the way we develop today or might be developing enterprise applications tomorrow. Among the speakers are:

- **People defining the J2EE platform and related technology.** J2EE spec lead **Mark Hapner**, Web Services JSR 109 Lead **Jim Knutson**, and other expert group members.
- **Major open source project committers/founders.** Apache Cactus founder and Struts committer **Vincent Massol**, OpenSymphony Group founder/core-developer **Mike Cannon-Brooks**, JBoss 4 lead architect **Bill Burke**, and others.
- **Authors of important enterprise development books.** More than 12 book authors including Agile/OO writer **Scott Ambler**, Core J2EE Patterns author **John Crupi**, Mastering EJB author **Ed Roman**, **Mark Grand**, and others.
- **Independent Evangelists and influential research analysts.** TheServerSide.com creator **Floyd Marinescu**, Java Lobby founder **Rick Ross**, patterns guy **Kyle Brown**, Giga Information Group VP of Research **Randy Heffner**, web services strategist **Anne Thomas Manes**, and others.

Why You Should Register Now!

TheServerSide.com has over 250,000 members so we expect this limited attendance event of 500 people to sell out quickly.

Date: June 27-29 Weekend, 2003

Location: Sheraton Colonial Hotel & Golf Club • Boston, MA

Who Should Attend: Java Developers, Architects, and Technical Managers who are looking for further insight into J2EE, XML, Web Services, Agile Methodologies, Extreme Programming, and Open Source Software.

Limited Attendance: Attendance is capped at 500 - you must act quickly to ensure your spot at the Symposium.

Registration Information

For more information visit <http://www.theserverside.com/symposium?sys1> See you in Boston!

TheServerSide.com
YOUR J2EE COMMUNITY

TheServerSide.com

Stay competitive in today's economy with the latest knowledge!

Network with top experts, forming relationships that will help you down the road.

Registration is limited to 500 people!

Get advice from the world's leading authorities in J2EE on critical decisions your projects are facing.

Improve your J2EE projects with newfound best practices.

We look forward to seeing you there!

From Within the Java Community Process Program

Proposed changes to JCP focus on day-to-day activities



Onno Kluyt

Welcome to the June edition of the JCP column! Each month I provide news and information about the Java Community Process: newly submitted JSRs, new draft specs, Java APIs that were finalized, and other updates from the JCP. June means it is JavaOne time, and hence this column will discuss the conference as well.

The Only Standards Body with a Version Number!

While the change from JCP 2.1 to 2.5 in October 2002 focused mainly on legal aspects such as license types for RI and TCK and the ability to do independent implementations, the proposed changes that would form version 2.6 of the process focus on the day-to-day activities in the JCP. This effort is done via JSR 215. I like to call out three areas of focus: Community Review, Expert Group formation, and TCKs. The proposal is to turn the Community Review into a public review (i.e., accessible to all, not just members) and to move the ballot after the existing Public Review. The goal is, this will lead to the JSRs entering the draft review phase sooner and that those reviews will receive more feedback. The JSR also suggests creating an observer category for expert groups. This enables the spec lead to distinguish between active participants and those who just want to stay abreast of developments while keeping the size of the group manageable for the spec lead. And expert group discussions and design decisions would be visible to interested JCP members. JSR 215 also proposes to set minimum TCK requirements for each JSR. This should lead to practical guidance to spec leads about this mandatory activity and to more uniformity across the JSRs. You can read more about this and send in feedback at <http://jcp.org/jsr/detail/215.jsp>. The Program Office and

Executive Committee members are very interested in your views.

JCP @ JavaOne

The Program Office is organizing and hosting various activities and events at this year's conference. First, there's the Java Community Evening event on Wednesday, June 11. This is a joint event with the JINI and JXTA communities and, for the first time, the Executive Committees and the Program Office will be presenting JCP awards for Best Spec Lead and Most Innovative JSRs. The Program Office will also have a pod on the exhibition floor that will provide a great opportunity to meet us and discuss your favorite elements and perhaps least favorite aspects of the JCP. And finally, as part of the conference, there is a Birds-of-a-Feather session on the JCP.

New Developments in J2ME

Recently a handful of new JSRs were submitted by JCP members. JSRs 216 and 217 propose to update both Personal Profile and Personal Basic Profile to account for the introduction of the Project "Swing" technology in this space, which I wrote about last month. JSR 218 proposes to update CDC with enhanced support for small electronic devices that don't have or don't require a graphical user interface. JSR 219 proposes to provide similar updates to the Foundation Profile.

Three JSRs did not fare well in their respective ballots. JSRs 213 and 214 were voted down by the JCP ME EC members in their JSR Review ballot, while JSR 177 was voted down in the Community Review ballot. Their supporters are now preparing for their respective reconsideration ballots.

Another JSR was added to the collection of completed and final specifi-

cations. JSR 139, CLDC version 1.1, was declared final and is available for review and implementation.

News from J2SE and J2EE

The world of Java technology APIs for the desktop and server space enjoyed a relatively quiet month (a sign of many engineers getting ready for the JavaOne conference?). JSR 151 released a third Proposed Final Draft specification, while J2EE version 1.4 is getting nearer to final release. JSR 152, JavaServer Pages specification 2.0, has also released a third Proposed Final Draft containing various changes such as API changes, updates to the Tag Library Descriptor, I18N updates, and a few changes to the Expression Language. JSR 109, Web Services for J2EE, has entered a second maintenance review.

Ballot Voting Is Public

In this column I've spoken a few times about ballots. The Executive Committees vote three times during the life of a JSR on the JSRs assigned to them:

- *At the beginning:* JSR Review ballot
- *Halfway:* Community Review ballot
- *At the end:* Final Approval ballot.

These ballots are public. To see how EC members voted on a JSR's ballot, go to the JSR's page on JCP.org (e.g., <http://jcp.org/jsr/detail/215.jsp> for JSR 215) and click on the links in the status table on the page (in this example the link called "JSR Review ballot"). If voting "yes," comments are optional. An EC member is required to enter comments when voting "no" and encouraged to do so when abstaining.

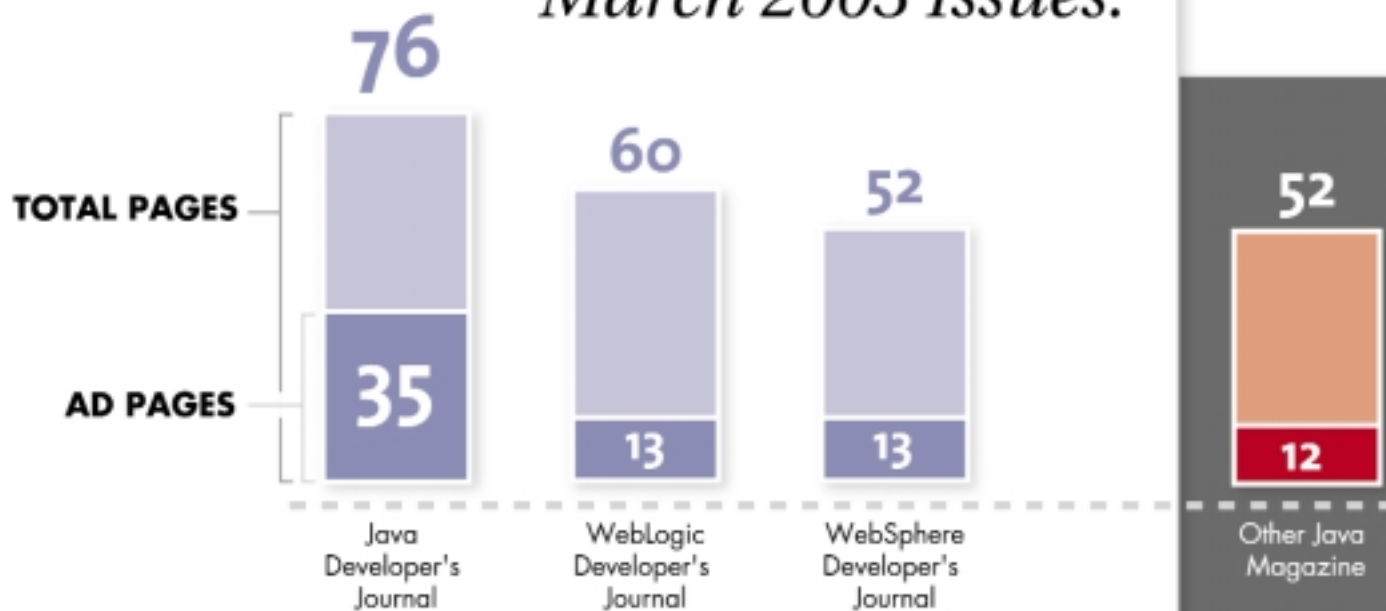
That's it for this month. I am very interested in your feedback. Please e-mail me with your comments, questions, and suggestions. ☺

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

onno@jcp.org

Java Served Here...

March 2003 Issues:



22%
...of all Java pages in March were published by the other Java magazine publisher

78%
...of all Java pages in March were served by SYS-CON Media



Carmen Gonzalez
Senior VP Marketing



Miles Silverman
VP Marketing



And the Artificial Stupidity Award Goes to...

I remember well the first time I worked at a company that used corporate e-mail. Instead of the usual development process that involved weekly meetings with users, between which we wrote specs and coded deliverables, this new messaging technology was going to streamline everything for us.

Unfortunately, the e-mail discussions became so fluid and nebulous that progress was paralyzed by the notes flying back and forth. Worse, a whole new type of project politics began to occur. Because your visibility was now measured in terms of the number of in-box entries you generated with influential people, second-rate employees who previously had little influence or peer respect could now inflate their image by sending notes and getting involved in discussions. It had been years since these media studies graduates had been able to contribute in any meaningful way to software development, and they collectively rued the day they had given up science for a softer study option because they failed to grasp basic euclidian geometry in the sixth grade.

After the rejection letters from law school and the local newspapers, they were forced to take up jobs as software testers, technical writers, or quality auditors. They were like a dormant cell waiting for the opportunity to wreak

revenge, and e-mail was their weapon. The ability to carbon copy notes made e-mail even more powerful to these Machiavellian employees, who now spent hours writing carefully crafted notes. The recipients of these notes were carefully selected so the writers could look good in front of them as they subtly put down peers with their superior grasp of corporate buzzwords.

During the entire morning that the corporate payroll leeches spent authoring their master memos, the rest of the useful project team were engaged in more mundane tasks such as writing and delivering code, assuming naively that deliverable solutions were what the company wanted. To the customer and senior management who were observers of the trail of back and forth e-mail, it became clear to them just who had things figured out on the project. "Let's see what Richard has to say about situation foo," because after all he had so much wisdom about other project issues, he must be the oracle to all solutions. When the whole project went belly up and the bean counters came in with their axes and marker pens, it was the programmers who were fired and the nothingware e-mailers who kept their jobs.

Alan Turing defined artificial intelligence as when you couldn't tell the

difference between communicating with a machine and a person. If the set of responses you got from either was indistinguishable, then the machine had managed to fool you. Each year IT departments at universities have a contest to build a real Turing machine.

I would like to propose a new contest, known as *Artificial Stupidity*. You write a program that sends unsolicited e-mail to announce its presence (a list of Dilbert cartoons or funny news stories is a good start), then it should monitor the e-mails it gets copied on and write a "Look I'm so smart" reply. This reply has to include buzzwords such as "prioritization" or "customer-focused," and preferably a few invented words as well like "responsivitation," "architecturalship," or "subliminablation." If you can get the phrase "business to customer electronic on demand commerce" in the reply as well, then you get awarded a bonus of 50 points and the Dan Quayle award, a gold-colored plastic potato mounted on a realistic granite plinth.

I suggested the idea of the Artificial Stupidity Award to a colleague at work and he seemed unimpressed. "We have some of those contest winners connected to the e-mail server already," was his reply. "They work in the human resources department." ♦



Advertiser	URL	Phone	Page
Altova	http://jdi.altova.com/wsd/		33
Asperon Corporation	www.asperon.com		20
Basis International	www.basis.com		79
BEA Systems, Inc.	dev2dev.bea.com/useworkshop		11
BlackHat	www.blackhat.com	916-853-8555	75
Borland Software Corporation	go.borland.com/j1		4
Canoo Engineering AG	www.canoo.com/ulc/		19
Colorado Software Summit	www.softwaresummit.com	800-481-3389	38
Crystal Decisions, Inc	www.crystaldecisions.com/cr9/218	800-877-2340	41
Empirix Inc.	www.empirix.com/know	866-228-3781	9
Ensemble Systems Inc.	www.ensemble-systems.com/glider	877-290-2662	53
ESRI	www.esri.com/mapobjectsjava	888-332-2320	37
Extentech	www.extentech.com/jdsale/		73
Fair, Isaac & Company	www.fairisaac.com/rules		Cover III
Fiorano Software, Inc.	www.fiorano.com	408-354-0846	65
Hewlett-Packard	www.hp.com/linux	1-888-hplinux	77
IBM Rational	www.rational.com/offer/javacd2		35
InetSoft Technology Corp.	www.inetsoft.com/jdj	888-216-2353	59
Infragistics, Inc.	www.infragistics.com	800-231-8588	14-15
InstallShield Software Corp.	www.installshield.com/puzzlejdi		47
iSavix	http://isavix.net	703-689-3190	81
JavaOne Conference	java.sun.com/javaone/sf		83
JetBrains	www.intellij.com		13
Jinfont Software, Inc.	www.jinfont.com/j6.htm	301-838-5560	31
Macromedia	www.macromedia.com/go/jrun4jdi	415-252-2000	87
Microsoft Corporation	www.microsoft.com/partner/empower		7
Morgan Kaufmann Publishers	www.mkp.com	800-545-2522	48
New Atlanta Communications	www.newatlanta.com		45
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	70
Oak Grove Systems	www.oakgrovesystems.com/jdj	818-880-8769	55
Oracle	oracle.com/experts	800-633-1072	17
Parasoft Corporation	www.parasoft.com/jdj2	888-305-0041	29
Precise Software	www.precise.com/jdj	800-310-4777	23
QUALCOMM Incorporated	www.qualcomm.com/brew		63
Quest Software, Inc.	http://java.quest.com/jclass/jdj		21
Quest Software, Inc.	http://java.quest.com/gcj/jdj		71
Quest Software, Inc.	http://java.quest.com/jprobe/jdj		Cover IV
RackSaver Inc.	http://opteron.racksaver.com	888-942-3800	69
Rational User Conference	www.rational.com/ruc		85
RealObjects	www.realobjects.com		39
RefactorIT	www.refactorit.com		42
ReportingEngines	www.reportingengines.com/info/ DJ _lune_ere.jsp	888-884-8665	24-25
ReportingEngines	www.reportingengines.com/info/ DJ _lune_ese.jsp	888-884-8665	57
ReportMill Software	www.reportmill.com/webstart	214-513-1636	43
Software FX	www.softwarefx.com		49
Sonic Software	www.sonicsoftware.com		Cover II
Sybase TechWave 2003	www.sybase.com/techwave2003		67
The ServerSide Symposium	www.theserverside.com/symposium?sys1		93
WebAppCabaret	www.webappcabaret.com/javaone.jsp		61
Web Services Edge West 2003	www.sys-con.com	201-802-3069	91
Ximian, Inc.	www.ximian.com/information/mgmt3		89
Zero G	www.zerog.com	415-512-7771	3
Zion Software, LLC	www.zionsoftware.com/products/buddy		51

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

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

Next Month

java.net.NetworkInterface: A Road Warrior's Friend

Since I can predict my downtimes, if I take my network card out, it's a safe bet I don't have a connection. Instead of tackling the rather large problem of detecting a good network connection, I took a different approach. I started researching whether I could tell my application that I definitely did not have a connection. If I could do that, it would be easy to write something that says "if not connected, then don't try to perform net-sensitive code," and that would take care of the majority of my downtime.

JSP 2.0 Technology: The Community Delivers!

JavaServer Pages technology originated more than four years ago as a powerful way to dynamically generate HTML on the server side. Over time, and with the input of the developer community, JSP technology has evolved and matured, keeping simplicity at the forefront. The next generation of JSP technology, version 2.0, represents an easy-to-use, robust, and extensible technology for building Web applications, well-suited toward generating dynamic Web content.



Avoid Bothersome Garbage Collection Pauses

Many engineers complain that the non-deterministic behavior of the garbage collector prevents them from utilizing the Java environment for mission-critical applications, especially distributed message-driven displays (GUIs) where user responsiveness is critical. How do we prevent these garbage collection pauses that interfere with the responsiveness of an application ("bothersome pauses")?



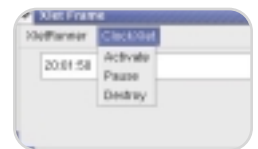
Xlet: A Different Kind of Applet for J2ME

In September 2002, Sun released the J2ME Personal Profile. Unlike the MIDP, which is the core technology for Java-enabled wireless phones based on Connected Limited Device Configuration (CLDC), Personal Profile is based on the Connected Device Configuration (CDC). The Xlet application model, which is inherited from the Personal Basis Profile, is one of its most important features.



ExtenXLS Java/XLS Toolkit 2.1 by Extentech Inc.

Extentech offers an intuitive, pure Java API for Excel integration. Under pressure from an anxious project manager, I evaluated it side-by-side with two other Java-based Excel integration tools available on the Web: POI (Apache Software Foundation) and JExcel. The requirements were for a fast, reliable tool that could push data from a Java-based application server to heavily formatted Excel templates in either Windows or Solaris operating systems.



Questions About Business Rules? Fair Isaac Gives You The Answers

Why bother with special software when I can just code IF statements in my applications?

Specialized rules software lets you store, find, and maintain large numbers of decision points quickly and easily. You get tools custom-designed for tracing and debugging logic flows. The rule engine is pre-written to handle the complexities of processing order. And unique maintenance facilities let you provide your business users with a safe and understandable way to control business logic without code or syntax.

Isn't a Java rule engine slow?

Fair Isaac Blaze Advisor is optimized to deliver great performance for different needs. It is the only rule engine on the market with selectable operating modes to choose from Rete-based determination of which rules to fire, sequential execution of all rules in a group, or compiled code execution for large-scale batch operations. And throughput is infinitely scalable by adding independent engine processing threads.

Can we manage large numbers of business rules?

Blaze Advisor lets you separate rules into discrete sets that are accessed only when called upon in your application process. You can segment rules based on functional task, maintenance needs, or authority levels for modification. Customers are using Blaze Advisor right now in applications with many tens of thousands of defined rules.

Is rules management software for developers or business users?

Blaze Advisor brings IT personnel and business users together with unprecedented cooperation. Technical Java developers can create rule architectures, process flows, and templates while business users populate the actual logic rules. By creating automatically-generated rule maintenance web pages, non-technical users can have controlled and secure access to view, create, or modify rules within the constraints that you set. And they don't ever have to see a line of code or a word of specialized syntax.

Will I have to install a proprietary database?

Fair Isaac Blaze Advisor has a flexible repository that works with your choice of flat files, databases, or LDAP systems. You can work with any JDBC database, XML document, Java class, or COM/.NET object in addition to building custom data interfaces to your systems.

Can I still use best programming practices?

Absolutely! Use strong typing on objects and properties. Do team-based development with versioning and check-in/check-out. Document your code with system-generated reports and your own comments on any object. Use built-in tools to set breakpoints, trace execution, and follow object references for impact analysis. And if you use the Rational Unified Process®, ask us for our free business rules plug-in!

Is it expensive?

You can get your feet wet for surprisingly little (call us and ask!). Then as your needs grow, you can add additional development seats, additional production CPU licenses, and even get enterprise licenses that bring the cost per user down so low that you can't afford not to use it!

Is it cost-effective?

Hundreds of companies (including more than half of Fortune's Top Ten) have gone through the purchase decision process and selected Blaze Advisor for their rules-driven applications. They are saving time and money with faster development of business logic, easier maintenance of decision points in their applications, better use of development resources by offloading business maintenance tasks to their business groups, and easier reuse of development work.

How do I find out more?

Send us an email at edm@fairisaac.com. We'll be happy to send you a demonstration CD, answer more detailed questions, set you up with an eval copy, or arrange a sales presentation. Have your cell-phone handy? Call us at 1-800-876-4900. And don't forget to check out our website at www.fairisaac.com/rules to download white papers or register for our informational web seminars.

ASK US...EDM@FAIRISAAC.COM



Change for
Tuesday

CUSTOMERS W/ A SCORE OVER ~~600~~ 590
GET A ~~PLATINUM~~ PACKAGE OFFER
SILVER

CAN YOUR CODE
KEEP UP WITH YOUR
BUSINESS GROUP?

Fair Isaac Blaze Advisor lets you create, deploy and modify the rules that drive your business. Maintained separately from the rest of your system code, business rules give you unprecedented visibility into and control over logical decisions used throughout your enterprise. Save time, save money, save yourself headaches with Fair Isaac Blaze Advisor business rules management. ***It's just a smarter way to do business™.***



www.fairisaac.com/rules

Introducing JProbe® 5.0

...now smarter and faster than ever

JProbe®

Find the cause of J2EE code performance, memory and threading problems faster than ever before with JProbe 5.0. New investigative features for finding memory problems combined with dramatic performance improvements mean even quicker resolution of problems in your application, servlet, JSP and EJB code.

JProbe Suite

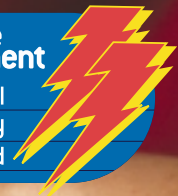
JProbe Profiler

JProbe Memory Debugger

JProbe Threadalyzer

JProbe Coverage

software
development
13th annual
productivity
award



See us at JavaOne!

Silver Sponsor, Booth #1501



For more info and a free eval, visit:

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