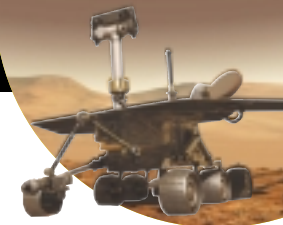




BRINGING MARS DOWN TO EARTH WITH **JAVA 3D**

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



▶ How to Explore Mars Without a Rover

**EXCLUSIVE INTERVIEW!**

# JDJ

WWW.SYS-CON.COM/JDJ VOL:9 ISSUE:6

HIGH TECH HAS  
**GROWN UP**

**Q&A WITH  
QUEST  
SOFTWARE  
CHAIRMAN & CEO**

**Plus...**

- ▶ Exposing J2EE Urban Myths
- ▶ An *i*-Technology Weather Report
- ▶ Where's Open Source in the App Server Surveys?

RETAILERS PLEASE DISPLAY UNTIL AUGUST 31, 2004

\$9.99US \$9.99CAN

07>



**VINNY SMITH**  
CHAIRMAN & CEO  
QUEST SOFTWARE



WILL IT WORK?  
Should you even  
have to ask?

Supports Oracle 10g

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

**Download your free evaluation today.** [www.datadirect.com/jdj](http://www.datadirect.com/jdj)

**DataDirect**™  
TECHNOLOGIES  
[www.datadirect.com](http://www.datadirect.com) 800-876-3101

Dralasoft is the technology  
of choice for business process  
management (BPM)

Technology that Xerox®  
chose to embed in DocuShare®

Technology that won  
Best of Show at AIIM 2004 in  
BPM/Workflow category

Technology that is enterprise ready

## Technology that *you* should consider

Dralasoft provides workflow  
processing as a component.

With a best of breed  
visual design tool.

Dralasoft Workflow™ is a  
comprehensive BPM solution.

We speak your language - Java, XML,  
SOAP, LDAP, J2EE.



# Did you know that leading software vendors embed Dralasoft Workflow?

## Can you afford not to put Dralasoft on the top of your list?

Learn more about who is using Dralasoft by  
visiting [www.dralasoft.com/customers](http://www.dralasoft.com/customers)

**dralasoft**  
*Workflow Automation Simplified*

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

Dralasoft, Inc. 10955 Westmoor Drive, Suite 110 Westminster, CO 80021 Email: [sales@dralasoft.com](mailto:sales@dralasoft.com) Phone: (303) 468-6754 Fax: (303) 468-6758 eFax: (303) 479-9836

Copyright 2004 Dralasoft Corporation. All rights reserved. Dralasoft Workflow is a trademark of Dralasoft, Inc. Java is a registered trademark of Sun Microsystems, Inc. Xerox and DocuShare are registered trademarks of Xerox Corporation.

# :\_œ`UfTZXe` `]d j`f`]]RTefR]]j f dVŽ

H VgV R]] S`fXYe`fc dYReV`WdYV]WRcVŽ DMUF TVU Sj  
gZ\_Z`d`WScRVeYc`fXY aVcWc^ R\_TV R\_U Vj Ma`aaZ\_X  
cVf]eLhV X]VW]]j ]`RUUV eYV]ReVc d`Wh RcvL`\_lj e`  
}\_U Ž Tf^ SVcd^ VR\_U`gVclj T^ a]ZReVUZ\_ U d`Ž  
h Rd d`TIVdW]]j Z`dR]]VU ŽŽ`\_ eYV dYV]VŽ


6\_Vclj d`Wh RcvE` `]dWc; RgR+ DZ` a]VŽ  
F\_`Seef dZVŽ\_2\_UcWdVYZ\_Xj f dWVŽ

2e6\_Vclj Lh VSM]ZgV eYRe eYVaf ca` dV` Wf W  
h Rcv e` `]d Ž e` YV]a UVgV]` aVcd TcVReV S VeeVc  
Raa]ZReZ`\_d WcVh Lh ŽY` fe XVeZ\_XZ` eYVh Rj`  
EYRe d h Yj` h VUVcZ\_VU`fc e` `]d e` Z`eVcReV  
dR^ ]Vd]j h ŽYZ`j`fc:56ŽEYVj ]Ve j`f R\_Rk  
lj kVj`fc T UML^`\_`Ž`c aVcWc^ R\_TML ZUV` eYV  
eYcVRUac`S]V^ dR\_U^ V^`g ]R]d` R]]h ŽY`fe  
ScVR\_Z\_Xj`fc d eZVŽ

Df cVh VTR\_`e`e`feC@: d f UZdLf dVc d` cZdLR U  
Raa]ZReZ`\_ ScZW eYReac`gV`fc d`Wh RcvZ`TcVkt  
Vd ac`Uf TeZgZj LcUf Tvd UVgV]`a^ V\_e Tj ]VdLR U  
cV]ZS]j`ac`Uf Tvd Vj Ma`aaZ\_X cVf]eLh Ž 3f e h VU  
acWc e` ]Ve j`f UcRh j`fc`h`\_ T`Tj f dZ`\_dŽ

D X`e`h h h Ž\_Vclj ŽT^`R\_U RUU 6\_Vclj e`j`fc; RgR  
UVgV]`a^ V\_e e`URj Ž:Wf c e` `]dU`\_eSVT^ VR\_ Z\_UZd  
aV\_dRS]VaRce`Wj`fc UVgV]`a^ V\_eac`Tvd LcVef`c`eYV^  
Wc R W]] cW\_U f\_UVc`fc 2\_eZDYV]WRcV 8f Rcr\_eVŽ  
9 Vj Lj`fc dYV]gVd RcvTc`h UVU V\_`f XY Rd Ž ŽŽ

GZV`fch VScVVe` ]VRc`^`cVRS`fe`fc; RgRe` `]d+

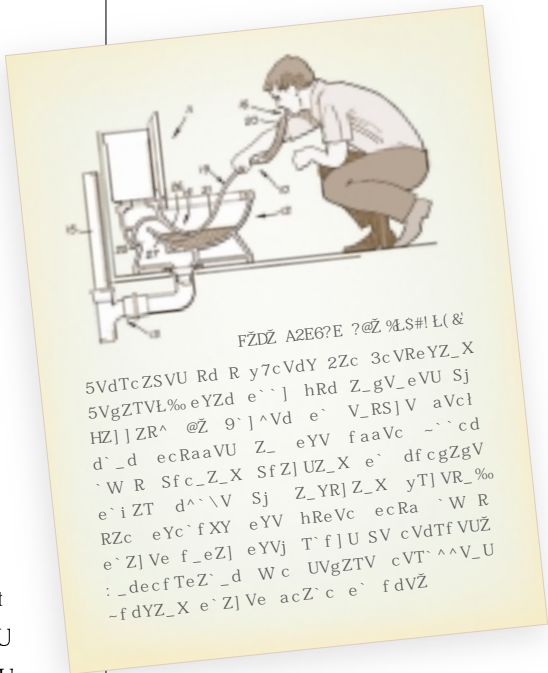
EVERJY |  CODE ANALYZER 3VbadRtZVdRf UZe` ]

EVERJY |  PERFORMANCE PROFILER :^ ac`gV Raa]ZReZ`\_ dWU

EVERJY |  MEMORY PROFILER CUFTVj`fc^ V^`g W eacZe

EVERJY |  THREAD PROFILER ZULj iLR URg`ZU eYdRU ac`S]V^ d

z`#!%6\_Vclj D`Wh RcvZ`] eYVed dVdGmLZ6\_Vclj R\_U 6\_Vclj D`Wh RcvRvcdRUV^ Rcd`W  
EVR^ d f UZL`\_TZ`RgrZReRcdRUV^ Rcv`WVY`\_>Zc`g dV^`dL`\_TZ



7c^`eYV vE`id J`f`cV  
?`e`=ZV]j 6gVc E` FdV%RcTYZgVZ

# EVERJY™

java software tools

h h h Ž\_Vclj ŽT^`

**Ad**



Jeremy Geelan

## An *i*-Technology Weather Report

Occasionally into any technology writer's life, a little rain must fall.

Sometimes of course it's not so much a little rain as a full-blown typhoon, such as when free and open source software (also known as FOSS for short) blows in as a development methodology.

Phrases like "all bets are off" come immediately to mind. We should all have guessed that it was going to mean stormy weather ahead when it took open source activist Bruce Perens numerous e-mails with his peers back in June 1997 to compile what was initially called "The Debian Free Software Guidelines" (referring to Debian, a distribution of Linux), but which eventually became shorn of Debian-specific references to become the "Open Source Definition" ([www.opensource.org/docs/definition\\_plain.php](http://www.opensource.org/docs/definition_plain.php)).

While we're on the subject of Linux distros, Red Hat founder Bob Young is a one-man hurricane. When he blew into an open source and free software conference held at the University of Toronto in May, he was scathing about the business model that fails to charge for software: "Good businesses will deliver more value to society than any nonprofit will," he gushed. "The profit motivation is actually a very good one; it makes sure we're delivering real value to our customers."

Contrast this with Professor Eben Moglen's comments at the same conference, during a panel discussion called "Free and Open Source Software as a Social Movement," and you begin to realize why the *i*-technology weather is so stormy, on a seemingly permanent basis. Everyone wants to be in charge!

"Whoever controls software, controls life," said Moglen, who is also legal counsel to the Free Software Foundation. He added: "Well, it had better be us. That's the real political meaning of the free software movement. Civil freedom in the 21st century requires human beings to retain control over the technological environment that surrounds them."

The subtext of the three-day event – as with any technology conference but most especially those concerning Internet technologies – was the future.

The future of the future, if you will. Of the IT future, anyway.

This is the one thing that unites every faction of the technology space: wanting to second-guess the shape of things to come. That's what everyone wants to know, in the hope perhaps of avoiding another dot-com boom-bust cycle. But that is the *only* thing that unites technologists of every stripe; once it comes to describing that shape, defining it and unpacking it so that IT organizations can prepare for it and move toward it, the tech community becomes a sometimes bewildering place, buffeted by winds from every direction.

Perhaps the answer lies in "The Grid" – that seems to be the hope anyway of Sun's Greg Papadopoulos, recently named as one of the 25 top CTOs of 2004, quite specifically for having taken the IT industry "one step closer to grid computing." Papadopoulos, a 20-year industry veteran and former MIT computer science professor, defines grid computing as "the decoupling of applications from specific hardware platforms." After this will come the virtualization of those platforms, Papadopoulos believes. This would allow IT organizations to create a network-based "computing and storage pool," and be a lot more dynamic in associating the computation they perform with the resources they have available.

If not grid computing, perhaps it is "utility computing" that will change everything. Embracing a shift from traditional infrastructure to utility computing, the IT gurus insist, "requires changes across three dimensions: people, processes, and technology." Certainly that is going to slow it down a tad, but you can see their point: a strong technical solution and architecture won't ever succeed without "buy-in" from application teams and sponsors for the changes in process. Any shift, whether it be to grid computing, utility computing, autonomic computing, or the Next Big Thing, requires a solid plan and organizational structure, otherwise neither application teams nor IT sponsors will benefit from reduced costs, faster time-to-market, or any of the other perquisites that accrue from a shared infrastructure. ☺

### Jeremy Geelan

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

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

### International Advisory Board

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

### Editorial

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

### Production

Production Consultant:	Jim Morgan
Associate Art Director:	Tami Beatty
Associate Editors:	Jamie Matusow Gail Schultz Jennifer Van Winkel
Assistant Editor:	Torrey Gaver
Online Editor:	Lin Goetz
Research Editor:	Bahadir Karuv, PhD

### Writers in This Issue

Reza Behrooz, Jason Bell, Jeremy Geelan, Michael Jacobs, Pramod Jain, Onno Kluyt, Satya Komatineni, Warren MacEvoy, Chris Moran, Joseph Ottinger, Kirk Pepperdine, Bill Roth, Greg Sporar, David Stephenson, Hani Suleiman, Bernhard Wagner, Joe Winchester, Geoffrey Wiseman

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

### Subscriptions

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

### Editorial Offices

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

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

### ©Copyright

Copyright © 2004 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission. For promotional reprints, contact reprint coordinator Carrie Gebert, [carrieg@sys-con.com](mailto:carrieg@sys-con.com). SYS-CON Media and SYS-CON Publications, Inc., reserve the right to revise, republish and authorize its readers to use the articles submitted for publication.

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

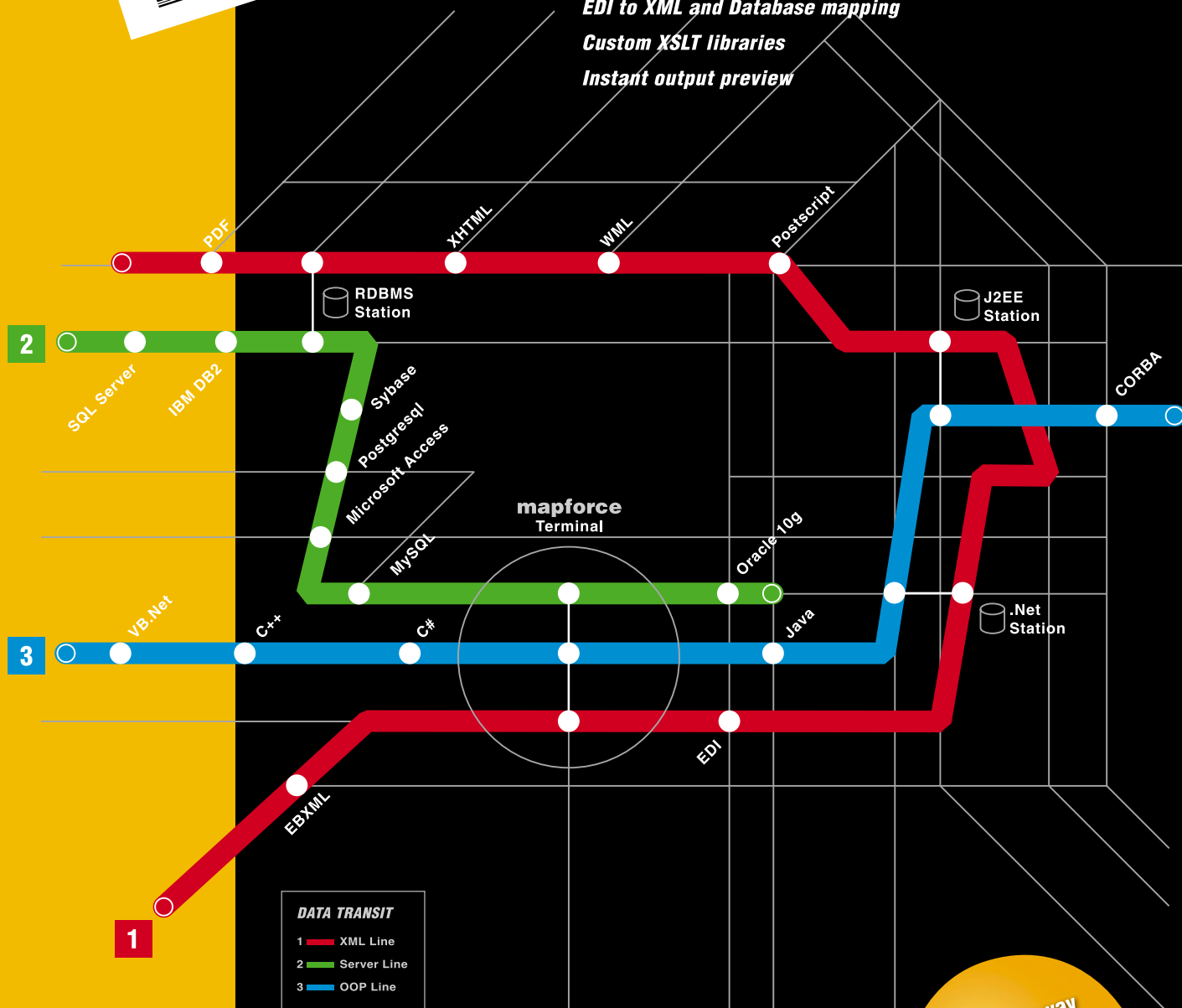
Newsstand Distribution Consultant  
Brian J. Gregory/Gregory Associates/W.R.D.S.  
732 607-9941, [BJGAssociates@cs.com](mailto:BJGAssociates@cs.com)

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





**New in RELEASE 4:**  
 XML to Database mapping  
 EDI to XML and Database mapping  
 Custom XSLT libraries  
 Instant output preview



**Next Stop, mapforce 2004!**

Make your way to the Altova booth at JavaOne for a live demo.



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

**ALTOVA®**

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



**Joe Ottinger**  
Editor-in-Chief



## In Search of Greatness

It's hard to find great Java applications. Next month's *JDJ* contains our Editors' Choice Awards, and so far for me it has the feel of a repeat – even though I decided to focus on applications I've been using day-to-day outside my own personal development environment. That's frustrating. I think that we – the Java developers – are creating programs and environments that are good across the board – but rarely great. I try to reserve my awards for those things that I don't consider “adding” to my development or deployment environment – they are my environment. They're the standard things I run without thinking about it.

That said, I tried to reevaluate everything from the viewpoint of someone who wasn't used to having my particular toolset at hand, by stepping back and saying, “Would I really like this tool so much if I hadn't been using it for so long? What would I think, coming in as someone unexposed to these programs?”

The results aren't encouraging. To be sure there are a few standouts, and a few programs that sort of come close, but picking three packages that need no qualification or justification is very, very hard. The problem, as I see it, is that Java creates applications that are typically above average...and that's it. To use a sports analogy, Java applications would almost always place in the top five, but rarely win the championship.

That doesn't mean Java isn't worth working with by any means. Placing in the top five consistently would be a marvelous thing, to an organization like the Atlanta Hawks or the San Diego Chargers, both typically woeful teams.

It still begs the question, though: How do we get Java over that hump, to create programs that really shine, not just from a user interface perspective but from the user's perspective?

Plenty of people have tried, and it's always getting better to be sure, but I don't know if the attempts made so far quite have the “it” Java needs.

We need to find a way to inspire passion for and with Java – apart from the invective hurled in all directions over SWT and Swing, JDO versus EJB versus Hibernate versus SDO, and Sun's licenses.

• • •

On other topics, I'd like to welcome Karl Avedal and Calvin Austin to the editorial troupe at *JDJ*.

Karl is a longtime acquaintance and one of the two founders of IronFlare, the company that created the Orion

Application Server. He left IronFlare a year ago and is now working on a research project on creating and using domain-specific languages. In addition to all this, he's served on several expert groups within the

J2EE field and has co-authored two J2EE books.

Calvin, no slouch himself, is the J2SE 1.5 specification lead and the lead engineer on Sun's Java on Linux port. He's been at Java Software since 1996 (before I even started using Java!) and co-authored *Advanced Programming for the Java 2 Platform* (Addison-Wesley).

Both of these guys are great assets to *JDJ* and I'm looking forward to working with them. I think they'll be able to contribute quite a bit to the magazine as a whole, and both have seemed happy to be able to serve in an editorial role.

Think long term, and enjoy. :) ☪



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

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

**SYS-CON  
MEDIA**

President and CEO:

**Fuat Kircaali** [fuat@sys-con.com](mailto:fuat@sys-con.com)

Vice President, Business Development:

**Grisha Davida** [grisha@sys-con.com](mailto:grisha@sys-con.com)

Group Publisher:

**Jeremy Geelan** [jeremy@sys-con.com](mailto:jeremy@sys-con.com)

### Advertising

Senior Vice President, Sales and Marketing:

**Carmen Gonzalez** [carmen@sys-con.com](mailto:carmen@sys-con.com)

Vice President, Sales and Marketing:

**Miles Silverman** [miles@sys-con.com](mailto:miles@sys-con.com)

Advertising Sales Director:

**Robyn Forma** [robyn@sys-con.com](mailto:robyn@sys-con.com)

Director, Sales and Marketing:

**Megan Ring** [megan@sys-con.com](mailto:megan@sys-con.com)

Associate Sales Managers:

**Kristin Kuhnle** [kristin@sys-con.com](mailto:kristin@sys-con.com)

**Beth Jones** [beth@sys-con.com](mailto:beth@sys-con.com)

### Editorial

Executive Editor:

**Nancy Valentine** [nancy@sys-con.com](mailto:nancy@sys-con.com)

Associate Editors:

**Jamie Matusow** [jamie@sys-con.com](mailto:jamie@sys-con.com)

**Gail Schultz** [gail@sys-con.com](mailto:gail@sys-con.com)

**Jennifer Van Winkel** [jennifer@sys-con.com](mailto:jennifer@sys-con.com)

Assistant Editor:

**Torrey Gaver** [torrey@sys-con.com](mailto:torrey@sys-con.com)

Online Editor:

**Lin Goetz** [lin@sys-con.com](mailto:lin@sys-con.com)

### Production

Production Consultant:

**Jim Morgan** [jim@sys-con.com](mailto:jim@sys-con.com)

Lead Designer:

**Tami Beatty** [tami@sys-con.com](mailto:tami@sys-con.com)

Art Director:

**Alex Botero** [alex@sys-con.com](mailto:alex@sys-con.com)

Associate Art Directors:

**Louis F. Cuffari** [louis@sys-con.com](mailto:louis@sys-con.com)

**Richard Silverberg** [richards@sys-con.com](mailto:richards@sys-con.com)

### Web Services

Vice President, Information Systems:

**Robert Diamond** [robert@sys-con.com](mailto:robert@sys-con.com)

Web Designers:

**Stephen Kilmurray** [stephen@sys-con.com](mailto:stephen@sys-con.com)

### Accounting

Financial Analyst:

**Joan LaRose** [joan@sys-con.com](mailto:joan@sys-con.com)

Accounts Payable:

**Betty White** [betty@sys-con.com](mailto:betty@sys-con.com)

### SYS-CON Events

President, SYS-CON Events:

**Grisha Davida** [grisha@sys-con.com](mailto:grisha@sys-con.com)

Conference Manager:

**Lin Goetz** [lin@sys-con.com](mailto:lin@sys-con.com)

### Customer Relations

Circulation Service Coordinators:

**Shelia Dickerson** [shelia@sys-con.com](mailto:shelia@sys-con.com)

**Edna Earle Russell** [edna@sys-con.com](mailto:edna@sys-con.com)

**Linda Lipton** [linda@sys-con.com](mailto:linda@sys-con.com)

*JDJ* Store Manager:

**Brunilda Staropoli** [bruni@sys-con.com](mailto:bruni@sys-con.com)





The right Java, whatever the gig.

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

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

[go.borland.com/j6](http://go.borland.com/j6)

Made in Borland® Copyright © 2004 Borland Software Corporation. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. • 21715.1

**Borland®**  
Excellence Endures™



Kirk Pepperdine

Java Enterprise Editor

## Where Is Open Source in the App Server Surveys?

Just recently Gartner reported that IBM has overtaken BEA in application server market share. The interesting thing is that Gartner's expression of market share is in a single number, dollars. While dollars are certainly an important factor in declaring a market leader, is this an accurate measure of market lead? If it is, where does that leave open source offerings such as Jonas and JBoss?

My past experiences have taught me to question claims or proclamations that one company has a lead in a particular market share. Take BEA for instance. When the application market was immature, BEA's WebLogic revenue figures included those revenues that were generated by sales of Tuxedo. Not exactly a fair measure for those companies that were only offering an application server at the time. Not that anyone expects BEA (or IBM for that matter) to be fair. After all, these companies know that people use these values to aid them in their decisions to ride with one vendor over another. As such, they understand that their survival depends upon those numbers being larger than their competitors'. When people buy into the illusion of success that is created by these numbers, they create a reality and in doing so reinforce the myth that the numbers have meaning.

The next question is, how does open source get reported? By definition, open source draws no licensing revenues and by definition carries a 0% market share. Is it just me or am I right in saying that it just feels wrong to be basing an important decision on a report that cannot accurately describe the usage of all the offerings in the space that is being considered? It leads me to question the validity of any conclusion drawn from a study that demonstrated a clear inability

to adequately model the space under consideration. But then again, we are talking about perceptions and perceptions can easily become reality.

To be fair, you cannot blame Gartner for this type of reporting. After all, it is this type of information that their customers are demanding. These customers will be making large investments into systems that are critical to their business needs. They need to know if the companies they are working with will remain solvent. In addition, they need to know if these companies will remain commit-



ted to the products that they are currently trying to sell. In their minds, the ability to draw in larger-than-life revenues means that a company will remain solvent and committed.

Take HP's entry into the application server market on October 24, 2000. "We'll form the core of HP's Internet offerings," said Kevin Kilroy, Bluestone's chairman and CEO. The buzz at the time was that this purchase would quickly propel HP into third place in the application server market. By June 2002, HP was reportedly in talks to sell the technology to Oracle. This can only be read as vindication to those who ignored the hype and took notice of Bluestone's 4% market share. The reality was that no one really under-

stood how HP could expand upon a 4% share under the economic conditions of the day. The question is, was that 4% value accurate and could the results have been different if there was a fair way to calculate market share?

It seems self-evident that if those people who were making the final purchasing decision limited themselves to only considering reports of market share, there would be no open source application servers in use. After all, who in their right mind would choose to base a business-critical application on a product that enjoys a 0% market share? Since they are in use, clearly their market share is not 0%. That said, how can you measure it?

There doesn't appear to be a good answer to the question. To the point, JBoss market share is reported to be 27%. How did BZ Media come up with the 27% figure? Again, I don't know how but here is a sample of their methodology for another study. "An e-mail was sent out to 10,000 randomly selected *SD Times* subscribers inviting them to participate in a study. Excluding bounces, the net number of surveys delivered was 9,792." How many of these responded is not mentioned. The fact that response is optional results in the survey being nonscientific. Sure, it's an improvement in that it appears to include open source, but is it any better or are the results any more reliable?

It's difficult to say if this survey would produce a more reliable measure of market share than revenues. What it does say is that our choice of application server almost seems random, which is what you'd expect to see in a commoditized market where you should not find any differentiating factors between competing products. If I recall correctly, this was one of the original objectives behind the J2EE. ☺

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

kirk@javaperformancetuning.com



## Royalties Fees are a Thing of the Past

Paying OEM royalty fees to application adapter vendors is now a thing of the past. Librados offers application adapters on a low-cost royalty free source code basis. This along with Librados' pure Java, J2EE standards-based architecture eliminates the proprietary vendor lock-in that has been a key factor in the high cost and limited adoption of old fashioned adapter solutions.

Librados delivers over fifty adapters for enterprise information systems such as SAP R/3, PeopleSoft, Siebel, Oracle Applications, JD Edwards, file systems, databases, and mainframe application environments. These adapters can expose back-end enterprise system functionality in a wide variety of formats, including XML and Web services.

© Copyright 2004 Librados, Inc. All Rights Reserved 10/04/04

**LIBRADOS**  
FREEDOM TO INTEGRATE

**SAP** Certified  
Integration

Call 888 341 4258 or visit [www.librados.com](http://www.librados.com)





**EXCLUSIVE  
JDJ Interview...**

Exclusive Q&A  
with  
**Vinny Smith**  
Chairman and CEO  
of  
Quest Software

Interview by Jeremy Geelan

**'HIGH TECH HAS  
GROWN UP'**

*A relative newcomer to the Java market, Quest Software's avowed mission is "to simplify IT management." JDJ asks Quest chairman and CEO Vincent C. (Vinny) Smith about J2EE, .NET, Web services, SOAs, the overall landscape for IT organizations, and the future of the technology space as he sees it – including the growth of the corporate Java market.*

**F**irst and foremost, thank you for agreeing to talk with JDJ, the world's leading *i*-technology magazine.

**JDJ: Why did Quest Software get into Java? Is this part of a bigger Quest game plan? Where do you plan to take your Java solutions?**

**Vincent Smith:** Java and J2EE are now an integral part of the IT technology stack. As a leader in application management, we needed to support our customers in developing and putting into production Java-based applications so that they can develop applications faster and manage them more effectively.

Our plan is to help make Java an even bigger success in enterprise applications with solutions that continue to improve developer productivity and application performance management. As Java and J2EE applications become more tied together with existing IT systems and service-oriented architectures, I think Quest Software is the company most able to provide integrated solutions that detect, diagnose, and resolve performance issues across all of these application components, regardless of platform or infrastructure vendor.

For example, our new Application Performance Management (APM) Suite for the J2EE platform enables companies to manage critical J2EE applications at every stage of the application life cycle with total confidence. It includes integrated application monitoring, real-time application server diagnostics, and system-wide J2EE diagnostics right down to individual lines of Java code. And it's heterogeneous, supporting all of the top Java application servers and platforms used by IT.

With our history of leadership in heterogeneous database development and management, we believe we are the only company that can offer IT this kind of freedom for Java with our solutions for Java-based application development and management. Providing new value to our Java customers will continue to be an ongoing focus for Quest Software.

**JDJ: Tell us a little about Quest Software's experiences with Java, compared to .NET. Do you believe there will always be room for both technologies?**

**VS:** I visit customer sites pretty often and you know, we haven't seen a lot of .NET, at least in large applications. I'm definitely seeing J2EE being used a lot in critical, high-volume enterprise applications. It's in those types of applications where management of performance and scalability is an issue.

I think J2EE and .NET will ultimately end up coexisting in the market. J2EE is great for large-scale enterprise applications, while .NET may be better suited for department-level applications and where rich GUI clients are a requirement. Businesses will choose the technology that best meets their needs for their applications.

**JDJ: Do you share the view that J2EE is over-complicated? Or is the latest generation of development tools easing the pain?**

**VS:** J2EE is certainly very complex – however, I don't think it's overly complicated. J2EE is designed to be a base platform for high-volume, scalable business applications. It's as complicated as it needs to be to achieve this goal.

But now that the base platform and underlying infrastructure of J2EE is there, it needs to be made easier to use – for developers, architects, quality assurance, and system administrators. Vendors like Quest are stepping up to make using J2EE easier and to help newcomers be productive faster. Industry groups like the Java Tools Community (JTC) and the Eclipse Foundation are doing their part. It's in everyone's interest to make J2EE more useful for the enterprise.

**JDJ: Will Java ever be challenged seriously in the enterprise, do you think, or on the server side?**

**VS:** Probably – there's nothing as constant as change in this business.

It's certainly not news that Microsoft has its sights set on the enterprise. We see it today in our database business with SQL Server. But I believe that .NET and Windows still have some maturing to do in the areas of security, transaction handling, and scalability before we will see it truly challenging Java for critical enterprise and customer-facing Web applications.

**JDJ: Take a moment to reflect on Java's history. What's one thing that Java has gotten completely right – and, conversely, is there anything in Java's history that you would like to change?**

**VS:** J2EE is a clearly a huge success. It has become the preeminent enterprise software platform in a remarkably short amount of time. In retrospect, more attention could have been paid to J2EE's complexity issues. But no technology is perfect. Back in the old days of "green screen" applications, there wasn't a lot between the application code and the OS, and those applications were also arguably less complex. Today, we have Internet architectures and the distributed J2EE technology stack, which enables much more complex applications to be built relatively easily. But, this also brings new and not-always-known dependencies to the system. I think the Java community is well on its way to addressing the issues of complexity.

**JDJ: When Quest acquired Sitraka, many developers were concerned as to the fate of favorite tools like JProbe, the first toolset for 64-bit Java – what's the status of JProbe today?**

**VS:** All of the products developed by Sitraka, including JProbe, are alive and well. I'm glad you asked this, because even though Quest is a relatively new name in the Java market, we have an impressive set of Java solutions that have, over the years, obtained tremendous reputations as leading Java tools. In fact, over 8,000 companies use Quest Java products today.

We recently released a new version of Quest JProbe with some big enhancements, including new features for investigating memory use. Earlier this year, we also extended JProbe's 64-bit Java support to Windows and Linux environments.

We just announced a major new version of Quest PerformaSure with new support for Oracle9iAS and BEA WebLogic JRockit, a new SQL browser for better application/database diagnosis, and other diagnostic features that our customers have been asking for. PerformaSure, along with JProbe, is a key component of our new complete life-cycle application management solution, the Quest Application Performance Management (APM) Suite for the J2EE platform.

Quest JClass also continues to be very popular with Java developers. We were pleased to see JClass ServerViews win a JOLT Productivity Award this spring. We are continuing to enhance the product line and you may see a new release by the time you read this. There's more going on that we could get into, but I think it's clear that Quest Software is fully committed to the Java market.

**JDJ: Sometimes acquisitions are done to eliminate competition. And sometimes acquired products stagnate in a larger company. Can you be more concrete about Quest's future directions for its Java products?**

**VS:** Our Java solutions fit very well with our overall mission to simplify IT management. Any technology stack that is key to the enterprise is important to Quest. We have made acquisitions in the Java space because of how important Java has become to the enterprise.

Quest Software's products and solutions are a reflection of the growing number of complex technology stacks IT relies on. Besides Java-based systems, IT organizations are tasked with managing complex custom application and database infrastructures, along with an extensive Microsoft-specific infrastructure (Exchange, Active Directory, etc.).

Although Quest does have products for managing Microsoft environments and database infrastructures, we think of Java as a distinct market and have structured the company with a dedicated team focused on delivering solutions to help manage the entire Java application life cycle. We have done this for Java application environments first, because that's where we see the demand and the growth opportunity.

**JDJ: Our readers are always interested in hearing about what people are doing with Java. Can you talk about some of the most interesting Java applications you've seen in production?**

**VS:** One interesting J2EE application I've seen is a Web-based song purchase application one of our customers, a major music retailer, has been building. It's along the lines of the online Apple Music Store. With J2EE they have been able to build a scalable, customer-facing application and put it into production in mere months. With the Quest APM Suite for the J2EE platform, we helped optimize the scalability of the system so that when it went live it could handle hundreds of simultaneous end-user transactions. Another Java application that many readers have heard about is the NASA Maestro software used to operate and visualize the data collected by the Mars Rovers. The application team used Quest JProbe to tune the Java code, and also used JClass components under the hood. The development team has said that they couldn't have put the project together as quickly or cost-effectively without tools like these. Very cool.

**JDJ: What are some of the biggest issues you see facing Java developers and managers today?**

**VS:** I see two particularly big issues for corporate Java developers.

The first is managing large-scale production J2EE applications. Ideally once an application goes into production, IT operations staff should manage it, but when a newly deployed J2EE application develops performance or scalability issues, the developers are usually needed (along with DBAs and other functional experts) to diagnose the problem. Even the most experienced J2EE developers have a hard time diagnosing issues in systems as large and complex as we have now. Intelligent, collaborative diagnostic tools really are necessary to diagnose and resolve problems quickly and keep critical J2EE systems running smoothly.

The second issue is dealing with increasingly heterogeneous Java infrastructures. Most companies are taking a "best of breed" approach in selecting Java and J2EE infrastructure, including the application servers, clustering, database back ends, etc. It's hard for one team to manage Java and J2EE application components when some run on WebLogic, some are WebSphere/DB2, and some are open source approaches like JBoss, Linux, and MySQL. Ideally, companies and developers should choose management and development tools that give them the freedom to use whatever J2EE infrastructure they like today, knowing that they can have the freedom to change it over time.

**JDJ: What do you think Java's role will be in the future of the enterprise applications?**

**VS:** Java really is becoming the core component here. Packaged

application vendors are adopting Java for new, modern user interfaces and new ways of accessing packaged and legacy applications.

For example, client-side Java-based GUIs provide a highly productive user interface for heavy users of a packaged application system. Less frequent or higher-level users of packaged apps have the option of server-side Web-based Java user interfaces, whether they're standalone or part of a "dashboard" portal set-up.

Beyond direct user interfaces, we see Java making it possible for packaged applications to be used in entirely new ways. I'm talking about Web services and service-oriented architectures. Rather than simply a user interface, Java can enable IT to build entirely new "composite" applications that use elements of their legacy packaged applications as service providers, data feeds if you will. This element is really changing the way that packaged applications are evolving, and it's really exciting.

**JDJ: What's going to be big in the next 18 months? Are there still new business opportunities in Java? Or should those wishing to invest in startup companies stick to investing in Microsoft's architecture and go with .NET instead?**

**VS:** Are there still new business opportunities in Java? You bet. But I don't believe individual investors should get into startups – that's the role of venture capital firms. Let me discuss the areas we are most interested in right now, that are most relevant to our customers.

First, there's the growing size of the corporate Java market. Java on both the server and desktop has matured to the point where now smaller and mid-sized companies are comfortable committing to it for their critical business applications. This increases the size of the market for Java solutions from development to diagnostic to application management.

This trend actually highlights one particular opportunity – the need to make development, tuning, and management tools smarter and more usable for companies new to Java. That might mean providing visual development tools with more expert performance tuning, or application management that works with adaptive infrastructure environments.

Then there are some emerging Java opportunities coming along, such as Web services, that have the potential to greatly improve business flexibility and productivity.

**JDJ: You actually started as an Oracle salesman. What advice would you have for someone young and starting out in the technology space today: should they be writing it, investing in it, selling it, or avoiding it?**

**VS:** You've got to go where your passion is. I'm passionate about making a real difference in how businesses use software to be more effective, and I followed that passion through good times and bad. It's the same for anyone starting out in high tech today – find the area that drives you and go for it. There's tons of room for innovation in IT today, but like any maturing industry, high tech has grown up. I visit a lot of companies and see their IT problems firsthand; IT will gladly pay for solutions, but they have to provide real value, real ROI. So high tech still needs fresh ideas and enthusiastic newcomers, in all areas.

**JDJ: Who would be the one client not yet captured by Quest that you'd be proudest to win over in 2004?**

**VS:** For our Java business? How about Microsoft? <grin>

Seriously, though, it's not the next sales win that I really get jazzed about, it's when a customer calls me to tell me that we made a big difference in their business. Or when I get an e-mail from a customer thanking our people for going that extra mile to help them really solve their technology problems.

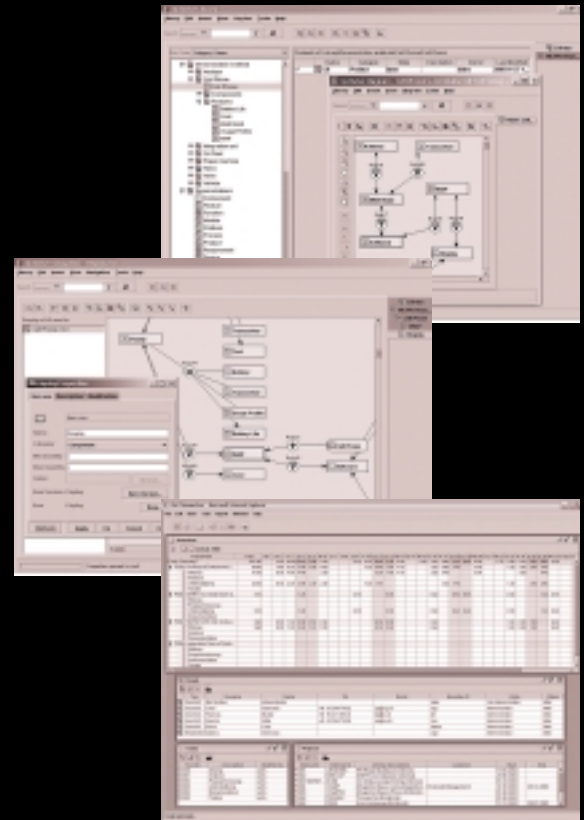
It's a testament to our great products and awesome, committed staff to hear this from our customers. And that's what makes me proud. ☺

# ULC

## Rich Thin Clients for J2EE

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

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



by Hani Suleiman

# Exposing J2EE Urban Myths

## The reality behind the legends

It has become fairly common these days when looking through blogs and various opinion pieces to hear a common cry: J2EE is a terrible, unwieldy, and cumbersome specification. While documentations from Sun and other vendors praise it, there is a lot of hostility and negativity toward it “down in the trenches,” so to speak. These trenches, of course, are populated with technologists who are always on the lookout for the next big thing, and the rank and file folks who look up to the technologists for wisdom and guidance.

In my opinion, many of these complaints are misguided, spread through rumor mongering and anecdotal stories with little to no effort made to validate them or place them in context.

Without further ado, I present you with nine urban legends surrounding J2EE.

### 1. *JNDI is awkward.*

JNDI is a lightweight directory service mandated by the J2EE spec that essentially consists of a directory of services where components can look up other components. Many people will speak derisively of JNDI; the latest buzzword compliance laws decree that you must use dependency injection to have components injected in rather than explicitly looked up. While both approaches have their merits and disadvantages, it's far from clear whether there is an overall winner.

JNDI is tremendously useful when you consider the fact that many vendor implementations will throw in features like a global clustered JNDI tree. Locally scoped JNDI references also free you from having to embed resource absolute names in your source code. JNDI is also fairly straightforward and easy to use; the barrier to entry is low enough that it really does not qualify as an obstacle.

### 2. *J2EE is difficult to develop in.*

While there is some merit to the

argument that EJBs and J2EE in general do not cater much to the in-vogue “agile” and “test-driven” development mentality, a lot of the blame should be placed squarely on vendors' shoulders.

For example, there are still vendors who do not support rapid hot-deploy, exploded application directories and runtime configuration. When having to work with such servers, it's not surprising that developers are turned off of J2EE and complain that it takes 20 minutes to preview every EJB change. It doesn't help either that many of these vendors had terrible implementations that performed abysmally and forced the hapless user to jump through any number of fire hoops to get things done.

The deployment descriptor hell is also a valid argument, but tools like XDoclet and the upcoming J2EE 1.5 greatly ease this pain. Going further, there's a blossoming market of tool vendors who try to cater to “rank and file” developers. WebLogic Workshop, WSAD, and Sun's Java Enterprise Studio are all fine examples of vendors stepping in to try to cater to a certain kind of developer. It's also important to consider the target audience for J2EE. A simple rule of thumb applies: if it doesn't fit, don't use it. Many people forget the “enterprise” aspect of J2EE. Don't overassume your project's needs and, even more important, don't overassume your own skill level. J2EE is not a trivial spec; it attempts to tackle some very thorny issues that, contrary to what you might have been told, do not have a simple solution.

### 3. *EJBs must be avoided at all costs.*

EJBs have been much maligned recently. Many loud and outspoken developers have made quite a bit of a ruckus over the inadequacies of EJBs, vowing never to touch them with a 10 foot pole. Unfortunately, in many situations this has resulted in a “let's throw the baby out with the bathwater” men-

tality. EJBs are being punished for bad marketing from Sun. No, they are not a silver bullet. In a lot of cases, entity beans are inappropriate and a huge overkill.

When it comes to session beans and message-driven beans though, there are clear benefits and advantages that should not be ignored due to the frightful aura that the letters “EJB” seem to have floating around them. Don't be so eager to dismiss the benefits of a component architecture that frees you from having to worry about pooling, security, transactions, clustering, and distribution. Part of the blame, of course, has to be placed on examples such as the early Petstore examples that encouraged the use of EJBs when it very often just didn't make any sense; but more on that later.

### 4. *J2EE is dead!*

In the trenches, EJBs on the sly have had and continue to have huge market penetration; while few people brag about this, nobody is surprised to find them slowly wriggling their way into many enterprises.

It's rare to find a financial institute that does not make extensive use of JMS. It's rare to find an enterprise that supports Java that has not purchased at least one J2EE application server. It's often more than a token purchase too, with important applications and services running on these servers. Also easy to forget is that a J2EE application will often not contain a single EJB (and be the better for it!). You're still using J2EE, and an application server will still provide your application with a lot of value-add. It's not a stigma to be bound to J2EE!

### 5. *J2EE is unportable.*

From the outset, one of the principles of J2EE was that vendors were free (and encouraged) to have their own deployment descriptors that allow deployers to fine-tune an application's

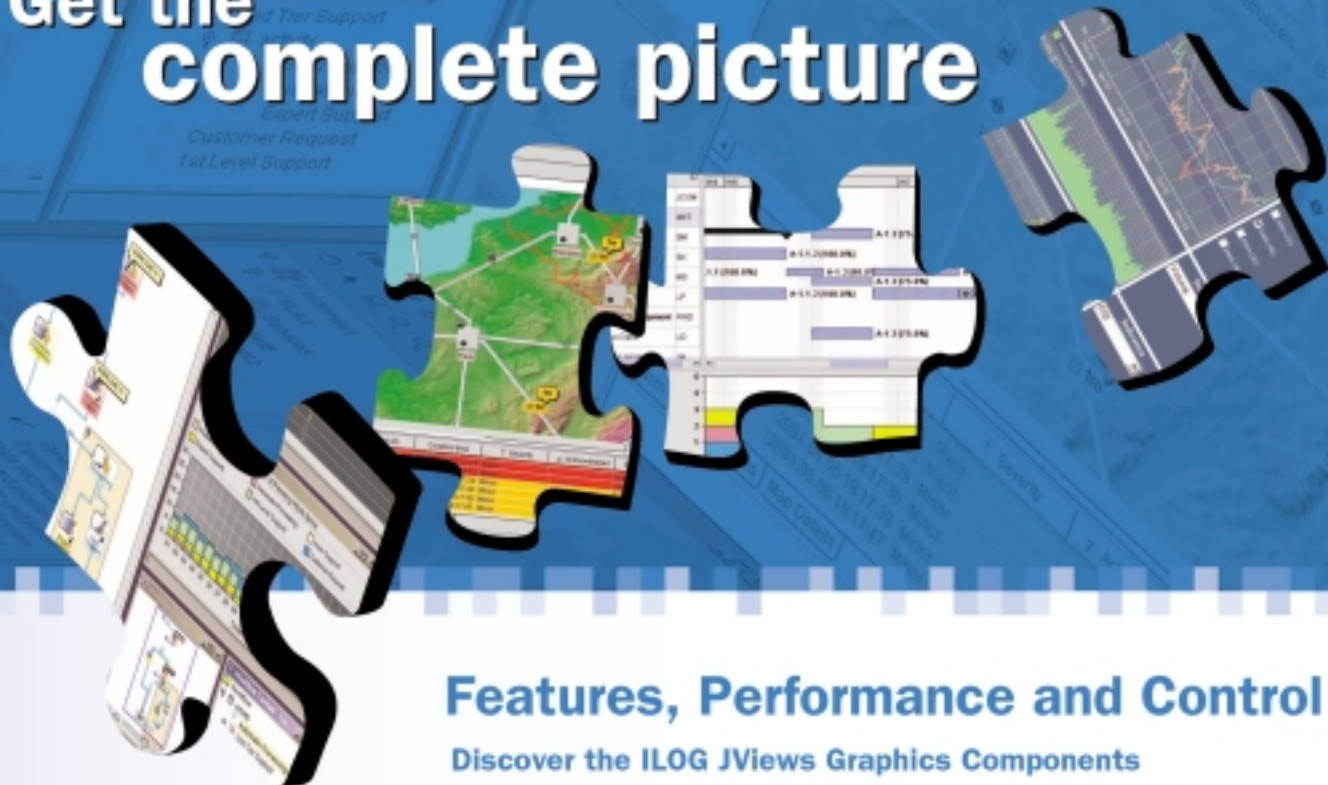


Hani Suleiman is the CTO of Formicary ([www.formicary.net](http://www.formicary.net)), a consulting services and portal solution provider. He is also a developer on a number of popular open source projects.

[hani@formicary.net](mailto:hani@formicary.net)



# Get the complete picture



## Features, Performance and Control

### Discover the ILOG JViews Graphics Components

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

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

Quickly and easily build:

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

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



Changing the rules of business™

deployment. Many vendors foolishly implemented this by mandating their custom deployment descriptors before deploying.

Realistically, you *can* have your applications be portable. Expecting it to be a trivial matter of dropping in the .ear file is naive and akin to expecting a Windows-developed Swing application to work flawlessly on an OS X machine without a single misplaced pixel. Sure it's doable once you do it a few times, but it takes knowledge of both Swing implementations to get it right.

To be fair to Sun, there have been a number of JSRs to help ease this problem. JSR-77 and JSR-88 (J2EE management and J2EE deployment, respectively) both attempt to reduce the vendor variance in these two areas.

Part of the problem also lies in the confusion about J2EE roles. While in reality it might be one developer who develops, assembles, and deploys an application, it pays to keep in mind that those three tasks are performed by different roles. A deployer, for example, would not know about the code that is being deployed, but would be able to spend the time and effort required to ensure the application is configured correctly to be deployed into a particular container and environment.

#### 6. *J2EE is expensive.*

Yes, it's hugely expensive if you go with the big name vendors. Even though some have dropped their prices, don't expect to feel you've gotten a good deal if you go for the IBMs and BEAs of this world. There are a number of both cheap and free alternatives that work just as well (and in some cases, just as badly!) as the "big" players out there. Whatever your budget, you're likely to find a vendor who caters to it.

The hidden aspect of expense is the nonmonetary costs associated with it. Yes, you do need someone comfortable in a J2EE environment. For some of the free offerings, you certainly need to invest the time and effort it takes to get some of the more obscure features you're after to work. Do not neglect to factor in the cost of the time and effort you'd be wasting by going to a third-party library and spending hours reconfiguring it for every new application.

#### 8. *Petstore is a reference implementation.*

Incredibly, many people use Petstore to "prove" that J2EE is overly complex and awkward. Petstore is not a reference implementation; it spends a lot of effort utilizing every approach and API possible. Taken as a whole, it's a collection of blueprints artificially packaged in one solution. Only a madman would use *all* the blueprints when developing his own app. Simply pick a few that make sense to you and use those, keeping in mind the project's requirements.

Sun seems to have realized the faults with Petstore and has probably come to regret the early hacked-together versions. The new Adventure Builder blueprints application seems closer to a "real-world" solution in terms of implementation, I'm told. If you really want a Petstore kind of example, look beyond Sun's to the myriad of implementations used to demo various frameworks and tools. With very few exceptions, they're generally better written and make a lot more sense.

#### 9. *J2EE is useless without extra frameworks.*

The first thing that most people do before embarking on any application, of any size or scope, is to try to use the chance to utilize any number of popular frameworks. In these days of framework mass hysteria, "naked" J2EE is perceived as being bereft of all functionality and quite useless out of the box.

Not so! While many larger applications would see a huge benefit in committing to a particular framework (be it open source or otherwise), smaller ones often do not merit the extra learning curve, maintenance cost, and configuration nightmares associated with many of these frameworks. Be flexible, consider your needs, and pick an intelligent solution that fits the problem at hand. If any of these frameworks was a true "one size fits all" solution, it'd be part of the spec.

This in fact does seem to have happened, with the introduction of JavaServer Faces. Unfortunately while well intentioned, the specification is far from adequate in its current incarnation. I am told that the expert group is aware of these issues and is hard at work addressing them.

“ Could it be that people don't know that every J2EE container is required to provide data sources that can be looked up?”

#### 7. *API "X" is too complex/unclear; it's easier to roll your own instead.*

It's amazing how people choose to do their own connection pooling, or try to manage transactions "manually." Could it be that people don't know that every J2EE container is required to provide data sources that can be looked up? Do they not know that every container supports pooling out of the box, that more often than not is incredibly trivial and straightforward to set up?

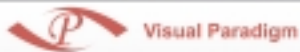
While it's enjoyable to tinker with various open source projects and play at integration with them, invest the time up front to use these features as provided by your application server. Yes, it's a lot less sexy; yes, it's a lot more boring. However, it will save time in both the short and long term, reduce your maintenance, and eliminate the "keeping up with the Jones'" aspect of tracking third-party sources. J2EE mandates many of these services, including but not limited to transaction management and connection pooling.

## Conclusion

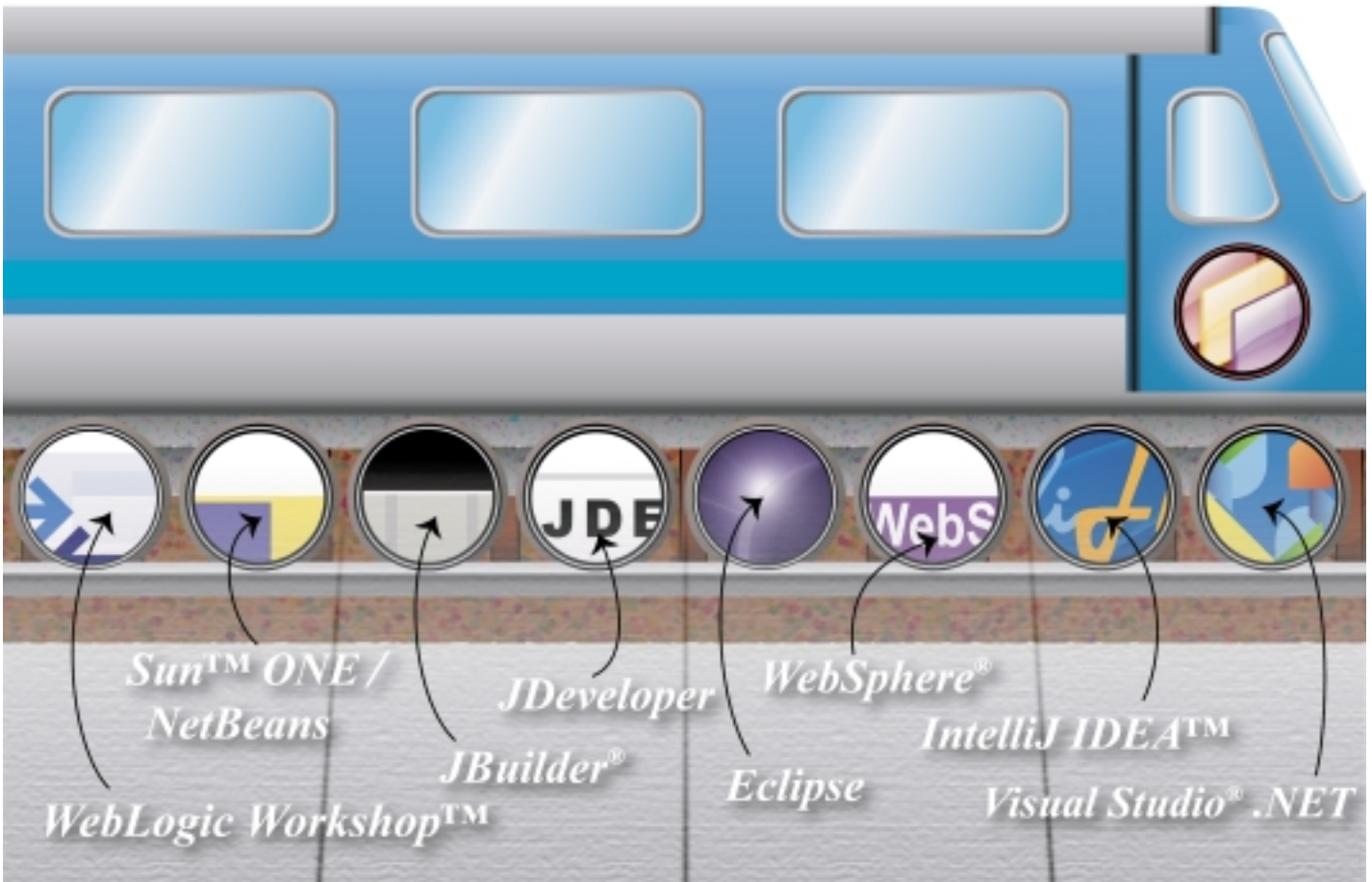
What's our take-home lesson? Put succinctly, take everything you hear with a grain of salt! It's often easy to get caught up in the hype (both positive and negative), but to date, nobody has come up with a solution that absolves us developers from having to think, consider, and apply our core skill to all our decisions: problem-solving in innovative and efficient ways.

J2EE is an evolving spec, albeit at a slower pace than some would like. It has areas it will mature greatly in, and despite many assertions to the contrary, it's a mature platform backed by a strong specification and huge industry commitment. Investing in it might no longer seem to be glamorous or innovative, but that's the sign of a mature and well-established specification. The technologists might have moved on, but for the rest of us enterprise application developers, we have a powerful and compelling tool in our arsenal that's useful far more often than not. ☺

# Model Once, Code Anywhere



## Interoperability Platform



**Build Quality Applications Faster, Better and Cheaper**

VisualParadigm  
for **UML**



VisualParadigm  
**SDE**



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

### Powerful features include:

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



Visual Paradigm

[www.visual-paradigm.com](http://www.visual-paradigm.com)  
[sales@visual-paradigm.com](mailto:sales@visual-paradigm.com)

# Managing Build Complexity with Apache Ant 1.6

*Anticipate and protect every possible security vulnerability*

by Geoffrey Wiseman

**A**s Apache Ant is applied to increasingly difficult tasks, its users are creating more complex and less legible build files. This is due, in part, to the limited tools for decomposition and code reuse within previous versions of Ant.

The AntCall and Ant tasks can be used to help manage complex build files, but they have a high overhead cost and are not always legible. Without stronger features designed for decomposition and code reuse, it's difficult to manage the increasing length and complexity of Ant build files, and attempts to do so can obscure rather than clarify intent.

Fortunately, Apache Ant version 1.6 has several features that can help you manage the complexity of your builds and make the files more natural, legible, and faster. In particular, Macro Definition, Preset Definition, and Importation are powerful tools that can help reduce complexity.

## Macro Definition

When generalizing functionality in Ant, we have historically been faced with a set of options, each flawed. You can duplicate the code (copy and paste), which creates maintenance issues and other problems. Alternatively, you can use the Ant and AntCall tasks, which allow you to run Ant targets in a parameterized way. Finally, you can create a custom Ant task, which is extremely flexible but time-consuming and complicated, when the desired functionality already exists as a set of Ant tags.

Macro definition in Ant 1.6 offers to change all that by allowing the Ant user

to create flexible macros that can be invoked within the Ant file without incurring the development overhead of a custom Ant task or the runtime overhead of an AntCall invocation.

If we compare the code in Listings 1 and 2, we can see that although the macro is a little less compact in this simple example, it's much more legible and will significantly reduce the overhead of code reuse by avoiding the reloading associated with an AntCall.

Macros seem even more powerful when you consider the ability to parameterize your macro with entire blocks of new code using the element tag, as shown in Listing 3.

## Preset Definition

Preset definition is a simpler, lighter, more compact form of a macro definition – the ability to customize existing tasks for your own purposes; it allows you to set default values or configuration settings for an existing task under a new name.

For instance, if you chose a more advanced logging structure for your build file, you might find it convenient to have an echo task customized to write a warning to your log file, as demonstrated in the following code:

```
1 <presetdef name="warn">
2   <echo file="build.log" level="warning" />
3 </presetdef>
4
5 <warn>Cannot locate configuration file:
   local.properties</warn>
```

By allowing you to customize and rename the task, Ant gives you the power to reuse common functionality and make the build files even easier to understand. These are important elements as everyday use of Ant becomes increasingly complex.

Listing 4 demonstrates how you might invoke a series of SQL state-

ments in Ant 1.5.

By way of comparison, Listing 5 demonstrates the marked improvements made by using Ant 1.6's preset definitions.

This is already a vast improvement. The preset definition is slightly more complex, but the overall legibility is markedly improved, and there is far less duplication. Further, the Import Task, discussed in the next section, could allow you to move the preset definition to another file that defines presets and macros.

## Import Task

Ant 1.6 comes with an import task for composing one build file from component pieces. Although it was already possible to do some of this using XML entity includes, the import task is in a lot of ways more powerful and less arcane.

In a lot of cases the import task will be used simply, such as:

```
<import file="project-macros.xml" />
```

It does have more complex uses, however. The new import task allows you to override Ant targets in an almost polymorphic manner. Imagine you had this simple build file as shown below:

```
1 <project name="SuperClass"
   default="hello">
2   <target name="hello" depends="sponsor">
3     <echo message="Hello, World!" />
4   </target>
5   <target name="sponsor" />
   An 'abstract' target -->
6 </project>
```

If you wished to make use of this previously developed functionality, you could use the new import task to override or alter the functionality in a way that would be very familiar to object-oriented developers. This is demonstrated in Listing 6.



Geoffrey Wiseman is a software consultant, one of Ingenura Inc.'s founders, and a long-time user of Apache Ant.

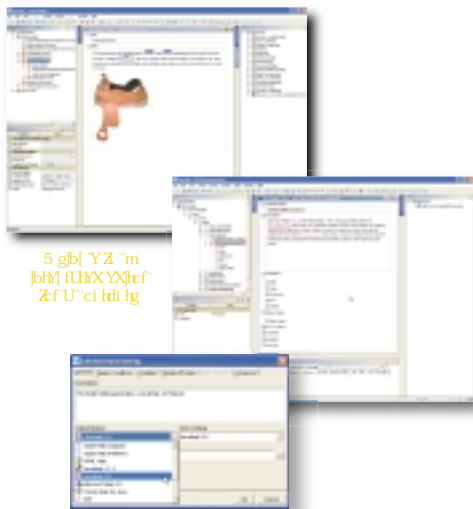
geoffrey.wiseman@ingenura.com

> 7ubDiga YcbYXYj Ycd'Ugjb[ 'Y'gci fVb[ 'U h'cf]b[ 'hcc`hUhgUj Yg'ha YzU`ck g' Zcf`W`UvcfUjcb`UbXfYXi Wg`\YUXUWYg'Zcf`XYj YcdYfg'UbX'hWb]W`k f]hYfg3

: ]bU`m'gca YcbY` \ Ug°  
 =bhfcXi V]b[ 'J YfYXi g<sup>IA</sup> · &'\$  
 G]b[ 'Y'Gci fVb[ 'A UXY'9Ugm



# VEREDUS



5'gb[ YZ`mi  
 ]b]f]hXYXhcf`  
 ZfU`ci hdi hg

Ci hdi h]g Ug'UgmUg Wccgbl  
 h'Yci hdi ha YXi a zgmY'UbX  
 k\YfY'lc g'cfY'hYci hdi h  
 lc`WUHY>U'U<Yd

5D='8cWa YbhU]cb

2`DfcXi W`XcWa YbhU]cb'i gb[ 'h'fa g'UbX'j cWm' UfmWa a cb'lc'k f]hYfg'UbX'Xj YcdYfg'

2`7fYUHY`XcWa YbhU]cb`U' hca U]W`miVmgWbb]b[ '>U' U'gci fW`WXY'UbXA]M'cgc'Zi  
 ' K ]bXck g'7CA`CV`Wg"

2`5i hca U]W`m[ 'YbYfUHY`g] bU' fYg'cZ5D='dfcdYfh]Yg'UbX'a Yh'cXgz'fYi fb'j U' Ygz'  
 ' dU'La YhYfz'UbX'k \ Yb`Uj U]U'VYZ'Yffcf`W'XYg'

2`FY]g'Ub Xj YcdYf`WXY'lc`a Yf[ 'Y'WUb[ 'Yg`]brc`nei f'JYfYXi g'dfc`Y'Wmei ]W`miUbX'YUg]h'z'  
 ' \_Yd]b[ 'XcWa YbhU]cb`U'W'fUHY'UbX'i d]lc'!XUHY'

2`; YbYfUHY`XcWa YbhU]cb`]b`Uj U]YmicZ'Z'fa Ug'Z'ca `nei f'JYfYXi g'dfc`Y'Wmg`W`Ug'  
 ' >U' U<Yd`UbXD8:'

G]b[ 'Y'Gci fVb[

2`5'gb[ 'Y'dfcXi W'h'U'WUHY'g`WbhYbhcbW`Zf`a i`h'd`Y'XY`j`YfU'Yg`]b`WbhYbhUbX`Z'fa`U'i

2`JYfYXi g'g] Yg'mi`'ha`Yz'fYXi`Wg`\YUXUWYg'UbX`WUHY'g'K`'7`Wa`d`]Ubh'ci`hdi`hZ'f`W'gca`h'X`'  
 ' `df]b]b[`ZK`Y'vcfUg'UbX`Uj`U]YmicZ<Yd`Z'fa`Ug'

<Yd`5i`h'cf]b[

2`Ei ]W`miWUHY`dfcZ'Ygg]cbU`\Yd`gng]Ya`g'UbX'i`gYf`a`Ubi`Ug`Z'ca`'Ug]b[ 'Y'gci`fW`'

2`7`Ub[ 'Y'nei`f`ci`hdi`h]Uf[ 'Y'nk`]h'ci`hid]bZ`'`Wbj`Yf]gcb`dfc`WggYg`''''

2`DfcXi`W`ci`hdi`h]b`U'bm'Z'fa`U'h`>U' U<YdZ`CfUWY`<YdZ`<HA@<YdZ`5dd`Y<YdZ`  
 ' `A]M'cgc'Zi<YdZ`K`]b<Yd`UbX`ch`Yfg`''

2`H'Ubg`UHY`h`Y`g'La`Y`]bZ'fa`U]cb`]brc`X]ZZ'fYbhi`Ub[ i`U]`Yg] ]U'L@: :`UbX`U`c`W`]hU]cb`dUf]bYf`

8ck b`cUX`U: F99'hf]U`cZF UgW`GcZk UfY`J YfYXi g`&'\$`  
 J ]g]h'k`k`k`"fUgW`gcZk UfY"W'a`#^X^cf`W`''&'\$\*!\*&(!+'`SS



As Ant build files grow and evolve, these kinds of capabilities will give Ant the power of composition, to keep modules of Ant code simple enough to understand while allowing increased overall complexity.

## XML Namespaces

While macro definition and preset definition allow you to make better use of existing Ant functionality, there are times when you will want to add entirely new functionality to Ant by either developing a custom task or making use of third-party tasks.

Because all Ant tasks and types have operated in a common namespace, there has always been the potential for conflict between Ant tasks libraries, causing unnecessary integration issues. The following code demonstrates how Apache Ant 1.6 reduces this through the use of XML namespaces, allowing two similarly named tasks to coexist simply by allowing each library the use of its own namespace through the new URI attribute of the typedef task.

```
1 <project name="NS-No-Conflict"
  xmlns:ingenura="http://alpha.org"
  xmlns:beta="http://beta.org">
2 <typedef resource="org/alpha/tasks.properties"
  uri="http://alpha.org" />
3 <typedef resource="org/beta/tasks.properties"
  uri="http://beta.org" />
4 <alpha:task />
5 <beta:task /> <!-- same task name -->
6 </project>
```

With the use of the new Antlib task, libraries can be bundled in such a way that they are imported automatically when the namespace is defined using the antlib-structured URI, as shown below:

```
1 <project name="NS-Antlib"
  xmlns:ingenura="antlib:com.ingenura.tasks">
2 <ingenura:task>
3 See how easy this is?
4 </ingenura:task>
5 </project>
```

## Looking Forward

Ant is being applied to increasingly

difficult tasks that call for increasingly complicated build files. Ant 1.6 provides several new and better ways to manage this complexity, making Ant files faster and more powerful, compact, and legible.

If you've been using Apache Ant, you'll appreciate many of the new features of Ant 1.6 that make it easy to use and enhance its power. If you haven't, perhaps this is a good time to take another look at Ant and what it can do for you.

## Additional Resources

For more information, consider the following resources:

- More information about the new features, including those found in the article as well as XML namespaces, Antlib, SSH, SCP, SubAnt and DefaultExcludes, can be found at <http://nagoya.apache.org/wiki/apachewiki.cgi?NewAntFeaturesInDetail>.
- Specific information about new and existing tasks can be found in the Ant Manual at <http://ant.apache.org/manual/>.
- For general information about Ant, see the Apache Ant Web site at <http://ant.apache.org/>.

### Listing 1: Using antcall

```
1 <target name="build.general">
2 <echo message="Building target ${build.target}" />
3 </target>
4
5 <target name="build.dev">
6 <antcall target="build.general">
7 <param name="build.target" value="dev" />
8 </antcall>
9 </target>
```

### Listing 2: Using macrodef

```
1 <macrodef name="mybuild">
2 <attribute name="buildtarget" />
3 <sequential>
4 <echo message="Building target '{@buildtarget}'" />
5 </sequential>
6 </macrodef>
7
8 <target name="build.dev">
9 <mybuild buildTarget="dev" />
10 </target>
```

### Listing 3: Using the element tag

```
1 <macrodef name="recordTask">
2 <attribute name="level" default="info" />
3 <attribute name="file" />
4 <element name="recordabletask" optional="no" />
5 <sequential>
6 <record name="@{file}" action="start"
  append="yes" loglevel="@{level}" />
7 <recordabletask />
8 <record name="@{file}" action="stop" />
9 </sequential>
10 </macrodef>
11
12 <target name="test">
13 <recordTask file="record.log">
14 <recordabletask>
15 <echo message="in log" />
16 </recordabletask>
17 </recordTask>
18 <echo message="out of log" />
19 </target>
```

### Listing 4: A series of SQL statements

```
1 <sql driver="org.hsqldb.jdbcDriver"
  sql="jdbc:hsqldb:\\ansible" userid="sa" password=""
```

```
  src="build/db/createUsers.sql">
2 <classpath>
3 <fileset dir="source/libraries" includes="hsqldb.jar" />
4 </classpath>
5 </sql>
6 <sql driver="org.hsqldb.jdbcDriver"
  sql="jdbc:hsqldb:\\ansible" userid="admin" password="admin"
  src="build/db/dropSa.sql">
7 <classpath>
8 <fileset dir="source/libraries" includes="hsqldb.jar" />
9 </classpath>
10 </sql>
11 <sql driver="org.hsqldb.jdbcDriver"
  sql="jdbc:hsqldb:\\ansible" userid="admin" password="admin"
  src="build/db/createSchema.sql">
12 <classpath>
13 <fileset dir="source/libraries" includes="hsqldb.jar" />
14 </classpath>
15 </sql>
```

### Listing 5: Using presets to reuse SQL calls

```
1 <presetdef name="runsql">
2 <sql driver="org.hsqldb.jdbcDriver"
  sql="jdbc:hsqldb:\\ansible"
  userid="ansibleAdmin" password="!ansibleAdmin">
3 <classpath>
4 <fileset dir="source/libraries" includes="hsqldb.jar" />
5 </classpath>
6 </sql>
7 </presetdef>
8
9 <runsql userid="sa" password="" src="build/db/createUsers.sql" />
10 <runsql src="build/db/dropSa.sql" />
11 <runsql src="build/db/createSchema.sql" />
```

### Listing 6: Using an import task

```
1 <project name="Intro-Import-Subclass" default="hello">
2 <import file="build-super.xml" />
3 <target name="hello" depends="SuperClass.hello">
4 <echo message="Powered by Ant 1.6" />
5 </target>
6 <target name="sponsor">
7 <echo message="Brought to you by JDJ." />
8 </target>
9 </project>
```

## Web Services Security

Since Web services can be accessed across the open Internet, security risks are inherent in Web service development. Depending on the type of Web service you are creating, an appropriate level of security will need to be implemented. The gamut can range from firewalls, to simple transport-level security such as basic authentication and SSL, to a variety of XML-level security mechanisms such as SAML.

At the very least, you must implement the proper security standards to ensure authentication, authorization, data integrity, data confidentiality, and proof of identity. One way to do this is to implement the WS-Security specification set forth by the Web Services Interoperability (WS-I) organization.

The WS-Security specification aims to define SOAP security headers and explain how they should be used. Developers should configure SOAP headers according to the following WS-Security specifications:

- Use security tokens to provide the server with client security evidence. Various tokens are available, and each contains different types of security evidence in various formats, which allows the message's target endpoint to verify client identity. For example, the Username token contains the name of the initiating client and an optional password.
- Use XML Signature to ensure that the message has not been modified. XML Signature is a standard that allows parts of an XML document to be digitally signed, thereby providing proof that the document has not been altered since the inclusion of the signature.
- Use XML Encryption to ensure that only the intended party can read the SOAP message. XML Encryption is a standard that uses cryptography to encrypt the SOAP message so that it is hidden from unintended viewers.

By following the standards set forth by WS-Security, you can create a more secure and reliable Web service, protecting the integrity and confidentiality of a message while also authenticating the sender.

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

**It's automated. It's fast. And it's the most versatile Web Services testing product ever.**



## Introducing Parasoft® SOAPtest®

The simple fact is, no other Web service testing product can do what SOAPtest can do. Or do it as fast and reliably. From ensuring functionality and interoperability to telling you how your Web service is performing, SOAPtest automates all critical testing processes across the entire lifecycle of your Web service.

**But don't take our word for it...** Go to [www.Parasoft.com/Soaptest](http://www.Parasoft.com/Soaptest) and try it for free – no commitment, no obligation. If you like it (and we suspect you will) just let us know. We'll be more than happy to help you and your development team get up and running.

For Downloads go to [www.parasoft.com/soaptest](http://www.parasoft.com/soaptest)

Call 888-305-0041 x3303 or email: [buzz@parasoft.com](mailto:buzz@parasoft.com)



### Features

- WSDL schema verification and compliance to standards.
- Automatic test creation using WSDL and HTTP Traffic.
- Data-driven testing through data sources (Excel, CSV, Database Queries, and so on).
- Scenario-based testing through XML Data Bank and Test Suite Logic.
- Flexible scripting with Java, Java Script, Python.
- WS-I Conformance: Basic Profile 1.0.
- WS-Security, SAML, Username Token, X.509, XML Encryption, and XML Signature support.
- MIME/DIME Attachment support.
- Asynchronous Testing: JMS, Parlay, Parlay X and SCP (SOAP Conversation Protocol) support.
- Identifies bottlenecks through SNMP, Windows, and JMX Monitors.
- Detailed Report generation in HTML, XML and Text formats.
- Real-time graphs and charts.

### Benefits

- Roll over uniform test suites from unit testing to functional testing to load testing.
- Prevent errors, pinpoint weaknesses, and stress test long before deployment.
- Ensure the reliability, quality, security and interoperability of your Web service.
- Verify data integrity and server/client functionality.
- Identify server capabilities under stress and load.
- Accelerate time to market.

### Protocol Support

- HTTP 1.0
- HTTP 1.1 w/Keep-Alive Connection
- HTTPS
- TCP/IP
- JMS

### Platforms

Windows 2000/XP  
Linux  
Solaris

### Contact Info:

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

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

by Bill Roth &  
Reza Behrooz

# The Theory of Innovation

## Turning J2EE into J2EE-easy with better tools and JSRs

**W**e know from the theory of relativity that the passage of time is relative to the perceiver. This is true of history as well. Sometimes history moves fast, e.g., during World War II and when communism was crumbling in 1989. Sometimes history moves slowly, as in the Cold War and the period between 1991–2001.

The same can be said of innovation. Sometimes a lot of innovation happens all at once as in the boom years of the Web from 1998–2001, and during the early days of Java – 1994–1995. Sometimes the pace of innovation slows to a crawl and other forces, principally economic, take precedence.

The J2EE platform is in the latter state at the moment. Much of the innovation has already been done, and most of the work being done now is “fit and finish” work. The net effect of this is that the application server vendors are innovating at a furious pace.

### History of the Development of the Platform

When the team at Sun was planning the original J2EE platform, it was clear that there were several stages in the process of developing a J2EE market, and that the industry had to be successful in order for J2EE to be successful. Furthermore, all the stages had to be complete in order for a fully functioning market.

First, the technology had to be adopted by large enterprise software companies such as IBM, BEA, and Oracle. JavaSoft’s unique product development methodology contributed to making this successful. The methodology, the precursor of the Java Community Process, was to include the “customer” – the application server vendors – in the development process. All of the vendors participated equally.

Second, the tools vendors needed to be included. It’s not sufficient to have

infrastructure available. It must be easy for developers to write code on top of the platform and do it easily, and there must be a vibrant commercial development tools market in order to accomplish this.

Third, there must be a market for software components so that it’s possible to assemble applications quickly using commercial development tools. Interestingly, this market did not develop as originally planned. For example, there are less than 30 EJB commercial components found on ComponentSource, a major site for selling software componentry.

Fourth, there must be a market for packaged applications, and these software packages could not have been written without the application servers, tools, or components available in the marketplace.

Finally, in order to build a complete ecosystem, a market for systems integrators must exist. In any technology market there must be people that companies can turn to in order to integrate the packaged software mentioned earlier. Once a need for this kind of integration is established, systems integrators become successful

and then a complete ecosystem has been established.

The key point of this model is that it is linear. Each stage must reach a certain level of success for any of the next stages to succeed. At this point in history, it’s safe to say that J2EE has a fully completed ecosystem, since all stages seem to be relatively successful. But there is one stage that has severely limited the further growth of the J2EE market. While there are successful tools in the marketplace like Eclipse and JBuilder, on the balance it is still very difficult to build, deploy, and manage J2EE applications.

### Fragmentation

This is further compounded by the fact that the major tool vendors seem to be heading in divergent directions. Both IBM and BEA are doing creditable jobs of improving the development process. However, they appear to be doing it in radically different fashions. IBM, for its part, has developed a popular open framework with Eclipse. Their commercial product, euphonomously named WebSphere Application Developer (WSAD), has extended Eclipse in many interesting ways, and seems for the most part to be hewing closely to standards. WSAD is a powerful, flexible tool, though all this power and the conceptual baggage that comes with it can be overwhelming.

From the vantage point of usability, BEA has done a great job of making WebLogic Workshop into a first class IDE in a mature market. It’s important not to underestimate what they have done. Their move in the space is akin to someone creating a new Detroit car company and succeeding. Workshop is essentially the Saturn of the development world.

BEA has always been an innovative company. In the past they have often implemented many Java features before the ink on the specifications was dry. They also have innovations



Reza Behrooz is an engineering manager at E.piphany and also serves on the Expert Group for EJB 3.0. He has over seven years of server engineering experience, mostly in J2EE. He holds a BSCS from Cornell University and a MSCS from Stanford University.

reza@cs.stanford.edu



Bill Roth is technical evangelist at E.piphany, a member of the *JDJ* editorial board, and Java editor for *LinuxWorld Magazine*.

br@billroth.net





Windows

Java

OS X

AIX

Solaris

HP-UX

OS/400

MSI

Mobile

Script

Linux

Redefining the Standard. *Everywhere*

# InstallShield X

Download Now! [www.installshield.com/redefine](http://www.installshield.com/redefine)

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

**InstallShield**  
[www.installshield.com](http://www.installshield.com)

of their own, which they've included in Workshop. Some of these innovations, especially the way they annotate code in the IDE, create code that is project-incompatible with any other tool, something I am sure BEA has considered. This is problematic, since a recent developer survey has shown that developers use on average at least two IDEs. For application developers, this makes our cross-platform and cross-application server development harder and lengthens our development cycles, something we would clearly like to avoid.

### Current Application Development Problems

There are other issues with developing J2EE applications. Most of these problems stem from the gap between the various J2EE components and the actual problems an application developer is trying to solve. Many components of J2EE are low-level APIs that, although they provide a great framework for building applications, still lack the high-

level and require all projects to reinvent the wheel and build the most basic elements of a UI framework, for example, component-based widgets or a standard navigation paradigm (Struts is designed to partly solve these problems). The second reason for the difficulty is the lack of tools that hide application developers from all the details of the APIs. For example, nearly all business applications require a UI for listing, searching, viewing, and editing a business entity that is often stored in a relational database. Ideally, building a simple application that performs these tasks should be fully automated with a WYSIWYG tool; in fact, the tool should also generate the necessary artifacts for data access (i.e., an EJB entity bean). Most applications often have at least several business entities, thus exacerbating the problem of building and maintaining such an application. Without the tools, the application developer is either forced to quickly write JSPs for each of these pages, which results in maintenance headaches, or to spend a significant

agement across application servers is nearly impossible. Furthermore, there is no integration between each of the core J2EE components and JMX. For example, there is nothing in J2EE on how JMS destinations should be managed at runtime. Without this, JMX and JMS are completely decoupled and thus each application server vendor builds its own set of MBeans and clustering functionality for JMS. Wouldn't it be better to define the common JMX MBeans for JMS destinations to enable standard tools for system management?

### Suggestions for Improvement of the Platform

From the current discussion, it's clear that several things can be done to make it possible to build better tools for J2EE. Given the current maturity of J2EE, it's important for new JSRs to try to move up and provide higher-level value-add features instead of the current trend of moving horizontally into new areas. We don't need new low-level APIs that try to tackle new problems as much as we

“ A good tool should make it easy for you to do simple things and still have the flexibility to accomplish difficult tasks”

level nature and ease of use needed to free application developers from building infrastructure. Tools can bridge this gap and make J2EE easier to use. A good tool should make it easy for you to do simple things and still have the flexibility to accomplish difficult tasks. Two areas that continue to exemplify some of these difficulties are building Web-based UI and system management.

Most applications currently provide a Web-based UI; however, such projects still involve employing J2EE experts who are familiar with several key APIs. Though JSP, Servlet, and Tag libraries, and some of the new JSRs provide a powerful framework, a new J2EE developer is still overwhelmed by the difficulty of building a page that shows some data. This problem is obvious if you compare the difficulty of building a J2EE-based UI page to the ease of building a similar page in Visual Basic or PowerBuilder. There are two main reasons for this difficulty. First, the current APIs are too low

amount of time in building a framework to provide tools and a higher level of abstraction over J2EE. The latter approach is used in most projects in which a few developers are tasked to build an internal framework to ease the application development by various methods such as code generation. These problems are not restricted to UI. As a result, most application vendors have a team of developers who build an infrastructure, often called a platform, and the necessary tools on top of J2EE to enable application developers to do just that – to develop applications.

Another problem area for applications is system management. This situation is analogous to the problems with UI. JMX has been adopted by almost all application servers as the standard for system management. However, there are no standards on the ontology of MBeans that an application server uses. Each vendor defines its MBeans with the desired attributes. As a result, system man-

agement across application servers is nearly impossible. Furthermore, there is no integration between each of the core J2EE components and JMX. For example, there is nothing in J2EE on how JMS destinations should be managed at runtime. Without this, JMX and JMS are completely decoupled and thus each application server vendor builds its own set of MBeans and clustering functionality for JMS. Wouldn't it be better to define the common JMX MBeans for JMS destinations to enable standard tools for system management?

### Conclusion

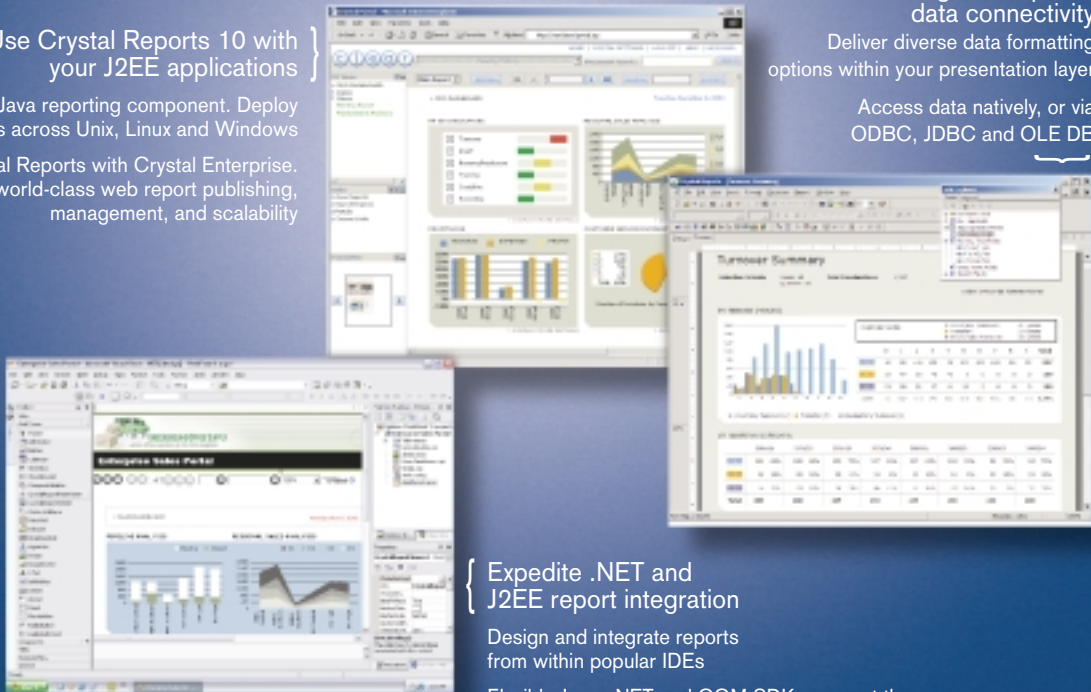
J2EE development is now at a crucial juncture. For J2EE to truly live up to its potential and reach a broader audience, it must become easier to use. While the feature set of core J2EE is now evolving slowly, the pace of innovation around the tools needs to speed up in order to effectively compete with .NET. ☛

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

Use Crystal Reports 10 with  
your J2EE applications }

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

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



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

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

{ Expedite .NET and  
J2EE report integration

Design and integrate reports  
from within popular IDEs

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



## New Crystal Reports 10.

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

# Developing Web Portals in Jetspeed Using JSP

Simplify the process

by Pramod Jain and Satya Komatineni

A Web portal is an application that aggregates multiple Web applications on a single Web page. Popular examples of portals are My Yahoo ([my.yahoo.com](http://my.yahoo.com)) and My MSN ([my.msn.com](http://my.msn.com)). These portals allow users to aggregate multiple Web applications (like Stock Quote, News, and Weather). In addition these portals allow users to personalize and customize the presentation and content of the individual Web application. This means users may do both: change the color, style, and layout of the page, and specify the content – list of stocks in a Stock Quote application and categories of news in a News application.

The Java community has created a standard for developing portal applications; it's called the Portlet API. The individual Web applications described above are called portlets. In August 2003, Portlet API specification, JSR 168 (<http://jcp.org/en/jsr/detail?id=168>), completed the public review process. The goal of the specification effort is to "enable interoperability between portlets and portals by defining a set of APIs for Portal computing addressing the areas of aggregation, personalization, presentation and security." This allows portlets to be developed once and deployed on any Web portal that is compliant with the portlet standard.

This article will describe how to develop JSP portlet applications using Jetspeed. Jetspeed (<http://jakarta.apache.org/jetspeed>) is an open source project from the Apache Foundation that provides tools to build Web portal applications. Its current release, Jetspeed 1.4, provides a portlet-based implementation. Jetspeed 2.0 is currently under development for release in early 2004. This will be compliant with JSR 168 portlet standards and built on Apache's Pluto, which is a reference implementation of JSR 168 (<http://jakarta.apache.org/pluto/>).

This article demonstrates a method to develop JSP portlets with a declarative specification of server-side logic instead of server-side programming. This is in contrast to current methods of developing JSP portlets that involve a significant amount of server-side programming.

The Jetspeed application is available as a WAR file and may be downloaded from <http://jakarta.apache.org/jetspeed/>. The WAR file may be deployed in Tomcat 4.1, WebLogic, WebSphere, or other J2EE application servers. The tutorial at <http://jakarta.apache.org/jetspeed/tutorial/> is a good guide to getting started. It explains how to aggregate, personalize, customize, develop, and deploy new portlets, therefore it will not be repeated here. This article focuses on developing portlets using JSP. To illustrate how JSP applications are developed in Jetspeed, we'll use a simple User Profile example (this will be referred to as Portlet 1 in Figure 1), with the following functionalities:

1. When a Web page with the User Profile portlet is loaded on the browser, the logged in user's address and phone number are displayed.
2. User changes the information and clicks on Update. The changed values are updated in the database.
3. The returned page displays the updated information.

## Standard Method for a JSP Portlet

An implementation of the sample application in Jetspeed, using the traditional method, will consist of the following files (see Listings 1–3). (Listings 5–7 can be downloaded from [www.sys-con.com/java/sourcec.cfm](http://www.sys-con.com/java/sourcec.cfm).)

- **profile.jsp:** JSPPortlet requires that the name of the submit form element have this prefix: "eventSubmit\_". The name after this prefix, for example, "doUpdate," is the method that is called in profileJSPActionClass.
- **profileJSPActionClass.java:** The action class must contain two methods – buildNormalContext and a method called from the JSP page. The buildNormalContext() method is the last method called every time the portlet is displayed or refreshed. If ProfilePortlet is submitted, Jetspeed calls doUpdate() first and then calls buildNormalContext().
- **profile.xreg (the Jetspeed registry file):** An XML file for registering a portlet, it specifies the type of portlet, JSPPortlet; the action class, indentJSPPortletAction, that will handle the submit event and deliver content to Jetspeed for rendering; and the startup parameters for the portlet.

Figure 2 shows the flow logic for four conditions in the action class:

- When the portlet application is first loaded in a Web browser
- When the portlet application issues a submit request that involves a database update
- When the portlet application issues a

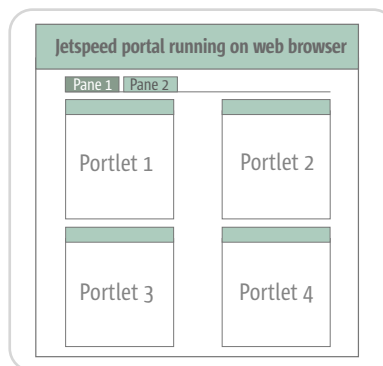


Figure 1 Layout of portlets in a pane in Jetspeed portal

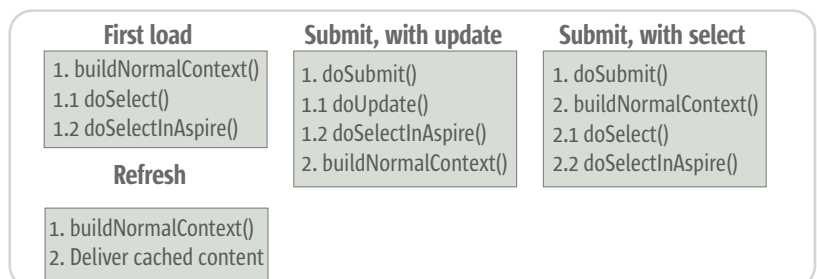


Figure 2 Flow logic



Pramod Jain is president of Innovative Decision Technologies, Inc. (INDENT) ([www.indent.org](http://www.indent.org)).

[pramod@indent.org](mailto:pramod@indent.org)



Satya Komatineni is chief technology officer of Innovative Decision Technologies, Inc.

[satya@indent.org](mailto:satya@indent.org)

# Javavavoom!

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

**Berkeley DB Java Edition**

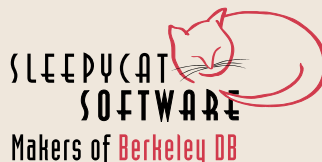
Download at [www.sleepycat.com/bdbje](http://www.sleepycat.com/bdbje)

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

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

**Experience the outstanding performance of Berkeley DB JE for yourself.**

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



submit request that involves getting data

- When the page containing the portlet application is refreshed or another portlet application on the same page is submitted

A Jetspeed container typically contains several Portlet classes – JSPPortlet, RSSPortlet, IFramePortlet, VelocityPortlet, etc. A JSPPortlet may be servicing several portlet applications – P1, P2, ... Pn. When a Portlet is submitted from a browser, the Jetspeed container determines which portlet class will process the request. The portlet class then hands over the request to the action class of the appropriate application. For each portlet application, developers write an action class and several methods in the action class. The action class typically calls a business component or database to satisfy the request. The data returned by the business component is used to render the return JSP page of the portlet application that was submitted. Other portlet applications on the same page are returned from the cache.

In the traditional method of developing JSP portlets (see Figure 3), developers write action classes and methods that are request specific. For example, if Portlet 1 has five JSPs and each has two

different submits, in a worst case scenario a developer would write 10 methods in Portlet 1's action class. Each portlet is typically configured to be processed by a different action class. Extending this analysis to four portlets with each requiring 10 methods would suggest that a developer would write 40 methods.

The issue with this approach is that for any reasonably sized Web portal, the collection of action classes and methods proliferates and becomes unmanageable.

### Declarative Method for Creating a JSP Portlet

Faced with this problem, we have developed a broker class that provides a single action class, IndentJSPActionClass, and a single action method, doSubmit(), to which all JSP pages are submitted (see Figure 4).

For brevity the broker class will be illustrated with an example in which action is limited to a database transaction. Two parameters are sent to the action class: actionType and actionName. actionType is "select" or "update" and actionName is the name of a request, whose purpose will become apparent soon. The broker action class calls Aspire, which is a declarative middle tier ([www.indent.org/aspire.htm](http://www.indent.org/aspire.htm)). Aspire is an open source J2EE pro-

gram developed by coauthor Satya Komatineni. To download Aspire's JAR file and the source code of the examples go to [www.indent.org/jetspeedDownload.htm](http://www.indent.org/jetspeedDownload.htm).

In Figure 4 JSPPortlet forwards the request to a single class and single method in the broker. Action Class calls Aspire with a parameter aspireURL. Aspire reads the declarative definition for aspireURL and transacts with the database. The retrieved data is transformed to a hierarchical data set (HDS). The outgoing JSP page then uses HDS to create the display.

The broker is employed in simplifying the action class. The role of the action class is twofold. In the case where "actionType=select", the action class is responsible for retrieving data from a database as an HDS. The outbound JSP uses the HDS for display in the browser.

In the case where "actionType=update", the action class is responsible for modifying content in a database. Once the update takes place, the user can be redirected to an appropriate page. Separating the role of the action class into these two categories offers clarity in the design of portlets.

The broker class has prebuilt parts that can construct an HDS declaratively from a set of SQL statements. This means the action class can make use of these definitions to automatically generate the necessary data for a Web page. In case of an update, the broker class provides a declarative update pipeline of parts where multiple update SQLs or stored procedures are executed against a database transactionally. By coupling these two declarative facilities, a generic action class for the JSPPortlet has been developed that streamlines the process for developing Jetspeed portlet pages rapidly. For more information about HDS and Aspire see "Using Hierarchical Data Sets with Aspire and Tomcat," [www.onjava.com/pub/a/onjava/2003/03/05/hds.html](http://www.onjava.com/pub/a/onjava/2003/03/05/hds.html).

An implementation of the sample application in Jetspeed contains the following files, shown in Listings 4–7.

- **Aprofile.jsp:** Uses Aspire and hierarchical datasets to get data for display. A hierarchical dataset is generated by Aspire and inserted into rundata in the action class. There are two methods, `getValue(column name)` that returns data for a column and `getChild(loop name)` that returns a hierarchical dataset containing multiple rows of data. This JSP page displays all the past phone numbers and addresses of this user and allows the most recent to be edited.
- **Java source for the Action class:** Uses Aspire, a declarative middle tier for all

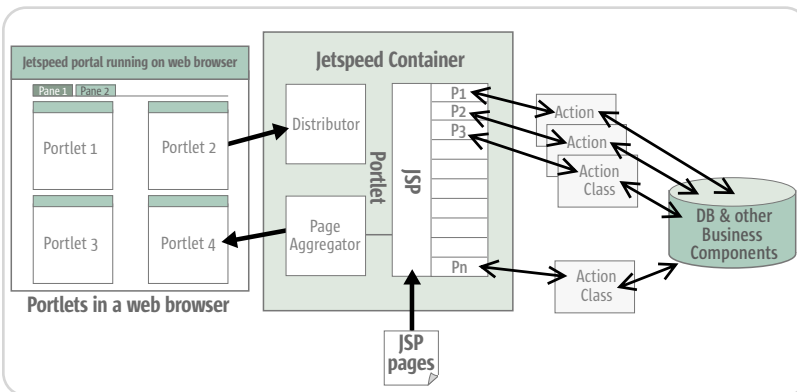


Figure 3 Schematic of the traditional method of JSP portlets in Jetspeed

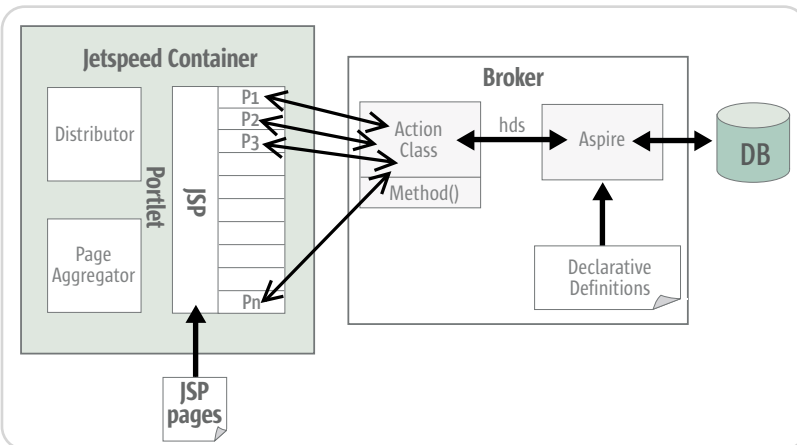
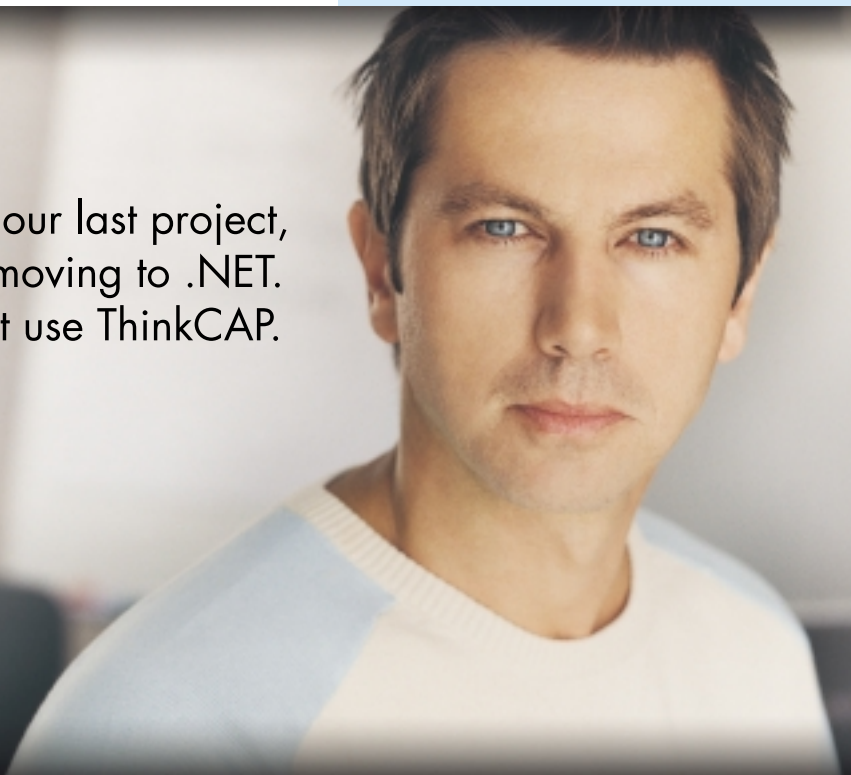


Figure 4 Schematic of components of Jetspeed and Broker

Think .NET development is more productive than J2EE?

Think **again**.

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



Think **better**.

**ThinkCAP**<sup>™</sup>

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

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

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

With ThinkCAP you can build quality applications faster.

Learn more about ThinkCAP at [www.clearnova.com/thinkcap](http://www.clearnova.com/thinkcap)

**CLEARNOVA**  
APPLICATION DEVELOPMENT • SIMPLIFIED • ACCELERATED

Highly Visual Development Environment

MVC Framework with Page Flow & Actions

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

Smart Data Binding<sup>™</sup> to data, objects, XML,  
sessions, or requests

Browser & server-side validation

Visual unit testing with RapidTest<sup>™</sup>

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

Content Management engine & tools

Supports .NET clients

Integrated, seamless security

Use any app server or 3rd party tool

database transactions; see Figure 2 for the flow logic.

- **profile.xreg:** Three additional parameters – `aspireAction`, `aspireURL`, and `appName` – have been added compared to the standard version. Values assigned to these parameters are used in `buildNormalContext()` to initialize the portlet and get the initial set of data from the database through `Aspire`.

- **Aspire properties file:** The database calls are declaratively specified in this file. `AspireURLs` illustrated here are “SaveAddress” and “addressURL.” `dprDB` is an alias for a database. In `SaveAddress`, the first part of request gets `user_id`, and the second part executes an insert. In `addressURL`, there are three parts to the request: the first two return a single row and the third returns multiple rows.

## Summary

A broker class is presented as a means of simplifying the process of creating JSP portlets. The custom Java coding-based approach in the action class is replaced with a declarative approach in which data access and update methods are specified in external properties files. This greatly simplifies the development and maintenance of portlets. ☺

### Listing 1

```
<%@ taglib uri='/WEB-INF/templates/jsp/tld/template.tld'
prefix='jetspeed' %>
...
<try
{ // rundata contains session data and data from action
class
RunData rundata = (RunData)request.getAttribute("rundata");
final String userid = rundata.getUser().getUserName();
String jspeid = (String) request.getAttribute("js_peid");
String address = (String) request.getAttribute("address");
<!--action URL is provided by the taglibs of Jetspeed -->
<form name="addressFrm" method="post"
action="<jetspeed:dynamicUri/>">
<INPUT TYPE="hidden" NAME="js_peid" VALUE="<%=jspeid%>">
<textarea name="address"><%= address %></textarea>
<!--submit button name must start with "eventSubmit_" -->
<input type="submit" name="eventSubmit_doUpdate"
value="Submit">
</form>
<% }
catch (Exception e)
{
AppObjects.log("Error:error in address.jsp page",e);
return;
}%>
```

### Listing 2

```
package com.indent.actions.portlets;
...

public class indentJSPPortletAction extends JspPortletAction
{

protected void buildNormalContext(Portlet portlet, RunData
rundata) {

public void doUpdate(RunData rundata,Portlet portlet)
{
    Hashtable args = new Hashtable();
    try
    {
        String address =
rundata.getParameters().getString("address");
        Class.forName("org.gjt.mm.mysql.Driver");
        Connection con = DriverManager.getConnection
("jdbc:mysql://localhost/document","root","indent");
        Statement stmt=con.createStatement();
        stmt.executeUpdate("insert into dummy (coll) VALUES
('address')");
        rundata.getRequest().setAttribute("address",address);
        stmt.close();        con.close();
    }
    catch(Exception e)
    {
        Log.error(e);
    }
}
}
```

### Listing 3

```
<?xml version="1.0" encoding="UTF-8"?>
<registry>
<portlet-entry name="Profile Demo App"
hidden="false" type="ref" parent="JSP" applica-
tion="false">
```

```
<meta-info>
<title>Profile Demo Portlet</title>
<description>Profile Demo Portlet</description>
</meta-info>

<classname>org.apache.jetspeed.portal.portlets.JspPortlet</c
lassname>
<parameter name="template" value="profileApp\pro-
file.jsp"
hidden="true" cachedOnName="true" cachedOnValue="true"/>
<parameter name="action"
value="portlet.indentJSPPortletAction"
hidden="true" cachedOnName="true"
cachedOnValue="true"/>
<media-type ref="html"/>
<url cachedOnURL="true"/>
<category group="Jetspeed">demo</category>
<category group="Jetspeed">jsp.demo</category>
</portlet-entry>
</registry>
```

### Listing 4

```
<%@ taglib uri='/WEB-INF/templates/jsp/tld/template.tld'
prefix='jetspeed' %>
...
<try
{
RunData rundata = (RunData)request.getAttribute("rundata");
final String userid = rundata.getUser().getUserName();
String jspeid = (String) request.getAttribute("js_peid");
String ckey=userid + jspeid;
boolean refresh_v=false;
String refresh = (String) request.getAttribute("appName");
if (refresh!=null && refresh.equals("profileApp"))
refresh_v = true;%>
<cache:cache key="<%=ckey%>" refresh="<%= refresh_v%>"
time="-1">
<%ihds hds = (ihds)request.getAttribute("PageData");%>
<FORM name="addressFrm" method="post" action="<jet-
speed:dynamicUri/>">
<INPUT TYPE="hidden" NAME="js_peid" VALUE="<%=jspeid%>">
<INPUT TYPE="hidden" NAME="appName" VALUE="profileApp">
<INPUT TYPE="hidden" NAME="aspireAction" VALUE="update">
<INPUT TYPE="hidden" NAME="aspireURL" VALUE="saveAddress">
<table>
<tr><td><textarea
name="address"><%=hds.getValue("address")%></textarea>
</td><td><input type="text" name="phone" value=
'<%=hds.getValue("phone")%>'> </td></tr>
<% ihds addressLoop= (ihds)hds.getChild("addressLoop");
if (addressLoop.isAtTheEnd()==true)AppObjects.log("Warn: No
data");
else {

for(addressLoop.moveToFirst();!addressLoop.isAtTheEnd();
addressLoop.moveToNext()) { %>
<tr><td><%=addressLoop.getValue("address") %>
</td><td><%= addressLoop.getValue("phone")%></td></tr>
<% } %>
<tr><td colspan="2"><input type="submit"
name="eventSubmit_doSubmit"> </td></tr></table>
<% } %>
</form>
</cache:cache>
<% } catch (Exception e){ AppObjects.log("Error:error in
jspage",e); }%>
```



Formula One e.Spreadsheet Engine

# EXCEL ENABLE YOUR JAVA APPLICATIONS

API-driven, 100% Pure Java toolset for embedding financial functionality into projects and applications



**BUILD JAVA REPORTS  
WITH NO LIMITS**

## ***Automate the building of Excel reports.***

Access raw data in databases, XML files, and other data sources. Perform calculations and formatting. Generate Excel reports with formulas, merged cells, charts, outlining, and other rich formatting upon delivery to users.



## ***Use Excel files to govern calculations and business rules in Java server environments.***

Java developers no longer have to translate Excel formulas to Java code. Spreadsheet experts (business analysts, actuaries, financial gurus, etc.) continue to own the core business logic and calculations. Calculation errors and application downtime is greatly reduced.



## ***Embed Excel-compatible data grids in applets and Java desktop applications.***

Users can manipulate data as if they are using Excel, including the use of Excel formulas and functions. After users make changes in your embedded grid, they can save the file to their desktop as an Excel file or commit the changes back to your server where they can be saved to a database or passed to another application.



**PRODUCTIVE DEVELOPERS • SUCCESSFUL PROJECTS • EMPOWERED USERS**

**ReportingEngines**

JAVA REPORTING TOOLS FROM ACTUATE

WEB: <http://www.ReportingEngines.com>  
EMAIL: [sales@ReportingEngines.com](mailto:sales@ReportingEngines.com)  
TEL: 913-851-2200 • 888-884-8665  
FAX: 913-851-1390

**FREE  
TRIALS**

### **FORMULA ONE E.SPREADSHEET ENGINE** (100% PURE JAVA TOOL FOR BUILDING FINANCIAL APPLICATIONS)

Excel reporting. Excel-like data grids. Server-side calculation and business rules engine.

### **FORMULA ONE E.REPORT ENGINE** (EMBED PDF, HTML, DHTML, AND XML REPORTS IN JSP AND SERVLETS)

Build reports against Java objects, JDBC, and XML. No report server to set up or maintain.

### **FORMULA ONE E.REPORT ENGINE FOR WEBLOGIC WORKSHOP** (ACCESS LIQUID DATA AS A DATA SOURCE)

All the power of the regular e.Report Engine as a fully integrated BEA Workshop Extension.

### **BEA PORTAL REPORTING SOLUTIONS** (POWERED BY THE FORMULA ONE E.REPORT ENGINE)

Visually build reports against BEA WebLogic Portal Server log files and BEA Liquid Data.



DESKTOP



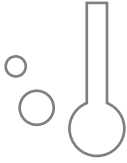
CORE



ENTERPRISE



HOME



# JFluid: A New Way to Profile Java Applications

Reviewed by  
Gregg Sporar

**A**nyone who develops production applications eventually spends some time profiling. JFluid is an experimental new technology for profiling Java code. It was developed at Sun Microsystems Laboratories and can be a handy tool in your profiling toolbox.

Your application should run fast and not overconsume valuable resources such as memory. For production applications it's important to "scale well" by running quickly and within a reasonable memory footprint when the workload increases. Profiling tools help identify bottlenecks in your code – sections that contribute the most to execution time and resource consumption. Profiling can reveal unanticipated facts about application performance. In the world of Java application development, all too often performance assumptions tend to be wrong.

Until now there have been two ways to profile Java code. You can modify the code by hand or use a profiling tool that leverages the Java Virtual Machine Profiler Interface (JVMPi). The first approach (e.g., inserting calls to `System.currentTimeMillis()`) is tedious and error prone. The second approach requires learning one of the many JVMPi profiling tools. Some are open source; others are commercial products. JFluid provides a third alternative.

## Features

First, an important note: currently JFluid works only with its own Java Virtual Machine (JVM). This JVM is a slightly modified version of Sun's v1.4.2 HotSpot JVM and is fully compatible with the unmodified JVM. The only difference is a small internal profiling API. Replacing the standard JVM with the JFluid JVM is easy – see the "Installation" section of this article.

The JFluid tool is a Swing application that is currently a bit rough around the edges – hopefully it will improve over time. The application to be profiled can be started by the tool. Alternatively,

you can attach the tool to a JVM that is already running. Attaching to a running JVM is useful when you want to profile applications that run in a container such as a Web or application server. An important JFluid feature is that no special command-line settings are required when starting the JVM to which you eventually attach. Until the JFluid tool is attached, the application runs at full speed with no profiling overhead. Think about the implication of that. It means that applications can be profiled in their normal deployed environment without incurring any overhead when profiling is not being done. In other words, JFluid allows profiling to be turned on and off at will.

Minimizing overhead is a cornerstone of JFluid's design. Its CPU profiling allows you to profile a subset of the application's methods. By not profiling the rest of the methods you can dramatically reduce profiling overhead. To profile a group of methods, first select one or more methods manually. JFluid treats each selected method as the root of a "call graph". Starting at the root method JFluid determines which methods are called by it, repeating the process recursively until the entire call subgraph is identified. Only the methods that belong to the call subgraph are instrumented and profiled; the rest of your code runs unchanged at full speed. Method selections can be changed arbitrarily while the program is running. This allows you to perform a "drill down" with decreasing overhead, or to successively profile different code areas for which there would be too much overhead if profiled together. Tools that allow only a selection of methods for profiling by name or package are unable to provide this functionality in such an easy-to-use form.

JFluid also tracks memory allocations with an eye on minimizing overhead. By default JFluid does not record complete information on every object allocation; instead it does statistical sampling by keeping a counter for

**Sun Microsystems Laboratories**

4150 Network Circle  
Santa Clara, CA 95054  
**Web:** <http://research.sun.com>  
**Phone:** 800 555-9SUN

each class. It decrements the class's counter when an object is allocated. When the allocation counter reaches zero, JFluid does detailed profiling and resets the counter. Detailed profiling of an allocation consists of capturing a stack trace and monitoring whether the object is still on the heap, in other words the object's liveness. By default the counter is 10 so 10% of object allocations are tracked. In production applications, particularly server applications, the number of objects allocated is so high that the information that is discarded is usually not significantly different from what is retained. The allocation counter can be decreased to improve precision or increased to reduce overhead.

For tracked allocations JFluid records the age of each object, where "age" means number of garbage collections the object has survived. It also reports the number of different ages for all tracked allocations of a class; this is called "surviving generations." The surviving generations value is useful for detecting memory leaks caused by objects that are constantly generated, but only partially garbage collected. In other words, the group of objects grows steadily. "Steadily" is the key word: it is not a group of objects that has been generated once within a short period of time in the past and remains fixed since then. Nor is it a group of objects that may grow for quite some time, but then get collected at once. In both of those cases, the number of different ages of objects within a group would be small, no matter how young or old the objects themselves are. It is typically in a leaking object group that the number of different ages grows steadily.



Gregg Sporar is a software developer who has been using Java since 1998. He is a staff engineer in the services division of Sun Microsystems and is a Sun Certified Java programmer.

[gregg.sporar@sun.com](mailto:gregg.sporar@sun.com)

JFluid can also tell you where in your code an object was allocated. However, it does not help answer the question of why an object has not been garbage collected. In other words, no display is provided to show which objects are pointing to the suspected leaking object. Sometimes just knowing where an object was allocated is enough of a clue to help you figure out why it has not been garbage collected. When that is not enough, a heap graph would be helpful. We have yet to see what JFluid is going to offer in this area.

In addition to the instrumentation of CPU usage and heap allocations, JFluid provides four monitors. These graphs show thread count, current heap size and usage, time spent doing garbage collection, and the count of the number of different ages for all heap objects. The count of the number of different ages is the surviving generations value for the entire heap.

### Installation

All the necessary .zip files are on the JFluid Web site. Support is included for SPARC Solaris, x86 Solaris, Linux, and Windows. Installation consists of two parts: replacing the JVM and installing the tool.

To replace the JVM, download the binary files (libjvm.so or jvm.dll, depending on your platform) and put them in place of the corresponding files in your standard JVM. Alternatively, you can download a complete JDK that has already been modified to support JFluid.

To install the JFluid tool, unzip the file into any directory. Edit the shell script or batch file that's used to start the JFluid tool so that your JVM and JFluid directories are specified, and you are ready to go.

### Sample Program

To demonstrate JFluid I wrote a simple prime number generator that uses the Sieve of Eratosthenes. Its method accepts a single integer parameter and returns all prime numbers less than or equal to that integer. (The source code for this article can be downloaded from [www.sys-con.com/java/sourcecec.cfm](http://www.sys-con.com/java/sourcecec.cfm).) The code is poorly written on purpose so that I can demonstrate some of JFluid's features. To test JFluid's ability to connect to a container, I also wrote a poorly implemented servlet that uses the prime number class; the servlet wastes an atrocious amount of heap space. Finally, I created a small .html form that invokes the servlet.

I ran JFluid on both a SunBlade 2000 running Solaris 9 and a Pentium3-based PC running Windows 2000. All screenshots in this article are from the PC, which is where I ran the example profiling session.

### Example Profiling Session

After modifying my JVM with the JFluid files, I started the Tomcat servlet container. After starting the JFluid tool I modified its set-

tings, which includes two different CLASS-PATH values. The first is for the main application class loader, the second for any other class loader that gets used by your code. For JFluid to be able to find and instrument classes of my servlet I had to specify the servlet's classes in the path for the other class loader (see Figure 1). In the Instrumentation tab of the settings dialog I turned on profiling of the core Java classes, which is off by default to reduce overhead.

To begin a profiling session I chose Run>Attach from the menu. JFluid prompted me for the name of the working directory for the JVM process – with my Tomcat installation this is the directory in which I started Tomcat. After specifying the directory I clicked on the console window that displayed when I started Tomcat; that allowed me to send the JVM a signal by pressing Ctrl-Break on my keyboard. I then returned to JFluid and verified that it connected to the JVM. This process is awkward but painless. The steps for connecting to the JVM running on a Unix system are a bit different; I have some tips on that below.

After JFluid attached to the JVM I clicked the Monitors tab to see the graph of heap usage. This graph is always available, even when no detailed instrumentation of memory allocations is being done. My next task was to look into the slow performance of the servlet. Using the Select and add root method> From binary menu entry, I specified my PrimeServlet.doPost() method.

With that done, I chose the Instrument> Selected root methods transitively from the menu and I was ready to go. In a browser window I brought up my HTML form, typed 123456 as the maximum value, and then clicked Submit. After a pause my browser displayed results.

Meanwhile, JFluid was profiling my code. Clicking on the Profile> Get latest results menu took me to the CPU results tab, which displayed the window in Figure 2. The top line showed that my doPost() method took 409 milliseconds (ms). Of that time, however, only 2.16 ms were actually spent in the code of doPost() itself; the rest of the time was spent in methods called by doPost(). In particular, the java.lang.String.split() method took up a huge amount of time – 243 ms. That's

almost twice the amount of time used by PrimeNumbers.getEratosthenes() to generate the prime numbers. Drilling a little deeper, I saw that most of the time in PrimeNumbers.getEratosthenes() was actually spent in a method I wrote to convert the answer to a string. Wasteful string processing was hurting performance.

In addition to wasting CPU cycles, the code was also wasting heap space. Using the Monitors tab to watch the graph of heap usage I noticed that the heap continued to grow as I used the servlet to request additional prime numbers. Some of that growth was expected, but the servlet was caching answers so requests with a previously requested value should not have caused much additional memory usage. But they did. This is due to a bug I put in on purpose: the cache was broken because the key used to add to the cache was not the same key used to request entries from the cache. So if 123456 is requested four times, there will be four instances of PrimeNumber in the cache, even though only one is needed. The extra instances of PrimeNumber cannot be garbage collected from the heap because the cache holds references to them.

To identify this memory leak I selected Object liveness from the Instrument menu. This turned off JFluid's CPU profiling and turned on detailed profiling of object allocations. Since this was a small application, I set

## Google Seeks Expert Computer Scientists

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

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

the allocation counter to 1 to get complete profiling. To reduce the overhead incurred, I set the stack trace depth that it records to 3. Then I went back to my browser and did four more requests of prime numbers.

When I clicked on Profile > Get latest results, the Memory results tab displayed with the list of objects allocated since my selection of Object liveness. The line of interest was this one:

```
4 live obj. - 4 alloc obj. - 3.8 avg. age
- 2 surv. gen. - 4 total alloc obj. -
PrimeNumbers
```

Each time I requested a prime number I used the same value: 123456. Only one live instance of the PrimeNumbers

class should exist. There are three extra objects because of the cache bug. After “live objects,” the other figures reported are:

- **alloc. obj:** The number of allocations being tracked
- **avg. age:** The average of the number of garbage collections the live objects have survived
- **surv. gen.:** The number of different ages of the live objects
- **total alloc. obj:** All allocations including those that are not being tracked

These numbers provide clues when you are looking for memory leaks. Double-clicking the line of figures brings up a window that shows the different stack traces that were

captured for each profiled object. In my example application the problem was obvious from the number of live objects reported. Unfortunately, memory leak detection is not always so simple. This is why the average age and surviving generation tail values are displayed. As described in the “Features” section, classes with large values for these figures are potential sources of memory leaks.

## Using JFluid on Unix Systems

The only difference between running JFluid on a Windows system and a Unix system is the method of attaching to a running JVM. On Unix systems the JFluid tool sends a SIGQUIT signal to the running JVM to establish a connection. To do that the JFluid client must be running with adequate privileges. For example, when testing on Solaris 9, I was attaching the JFluid client to the JVM used by Sun ONE Web Server v6.1. By default, that JVM was run by the user ID nobody. So I had to run the JFluid tool as nobody, not as another user.

## Conclusion

JFluid is an experimental but powerful tool. It provides detailed profiling information on demand, allowing you to turn profiling on and off at will. When profiling is on you can control how much overhead is incurred. The JFluid tool lacks some features and polish, but it's easy to learn and installation is quick. ☺

## References

- **JFluid:** <http://research.sun.com/projects/jfluid/>
- **JFluid research:** <http://research.sun.com/techrep/2003/abstract-125.html>
- **JVMPi:** <http://java.sun.com/j2se/1.4.2/docs/guide/jvmpi/jvmpi.html>

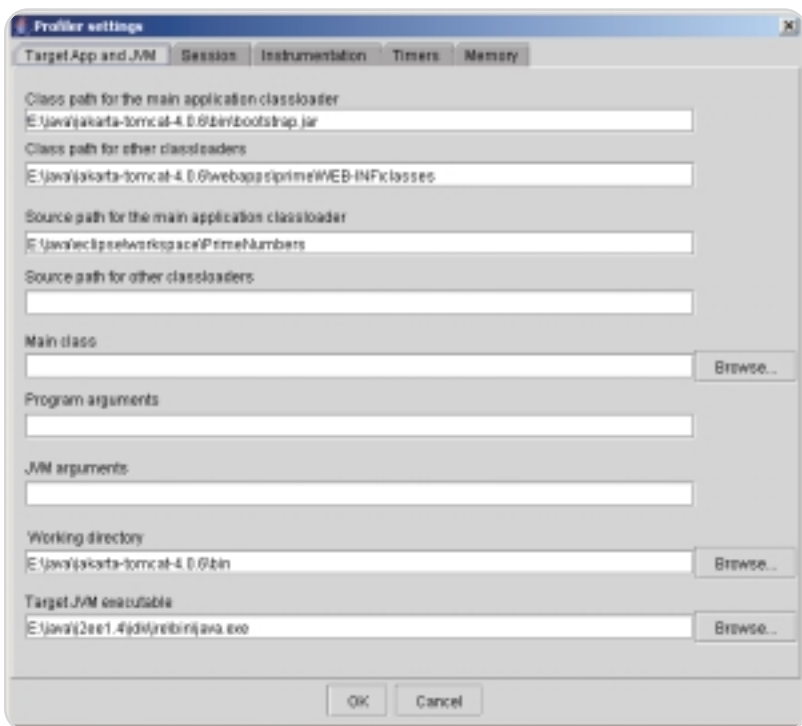


Figure 1 Profiler settings

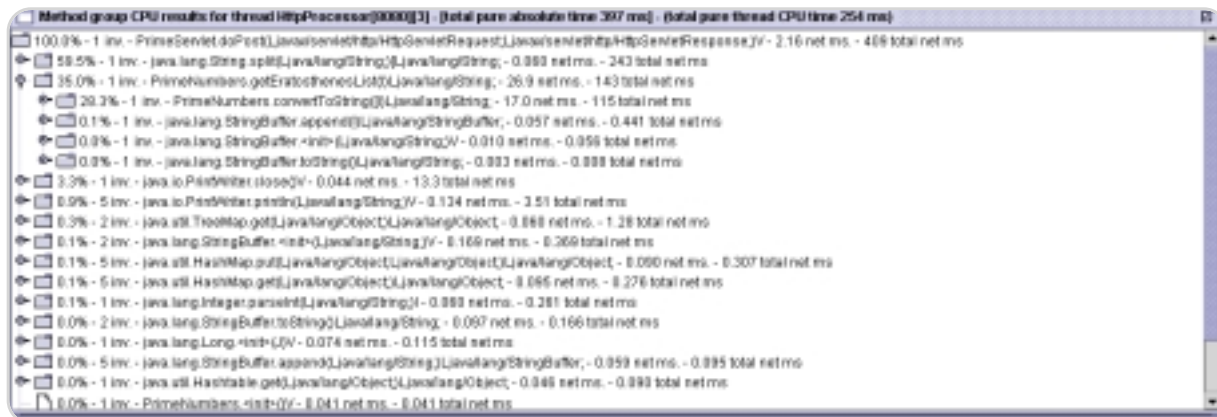


Figure 2 Method group CPU results

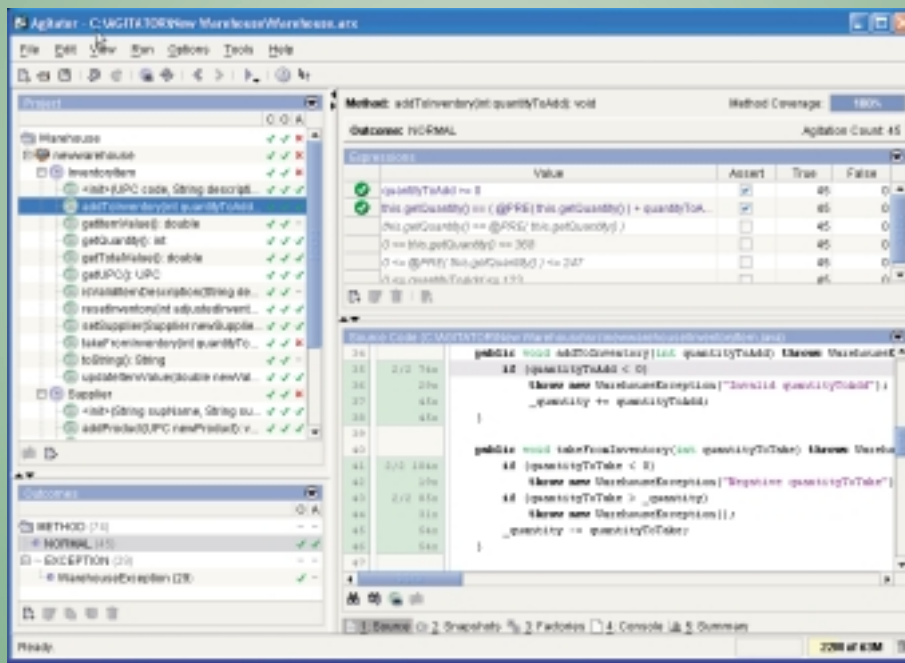
See us at JavaOne  
at booth #832, and  
attend Alberto Savoia's  
presentation "Beyond JUnit".

# JUnit started the Developer Testing revolution.

## Are you ready for what comes next?

A unique interface  
lets you know for  
each class and method  
when tests meet  
your criteria for code,  
outcome, and  
assertion coverage.

Automatically  
creates tests for  
all possible outcomes,  
including expected  
and unexpected  
exceptions.



Get extreme code coverage with automated test data generation and fully configurable test data factories.

Automatically generated observations give insight into your code's actual behavior and can be converted to durable unit tests with a single click.

Fully integrated code coverage to make sure you do not miss any code.

## Agitator™ - The future of Developer Testing for Java™

Agitator intelligently exercises Java code to show you observations about its behavior. You can convert valid observations to unit tests with a single click. This unique process, called Software Agitation™, helps you create thorough sets of durable unit tests. It lets you edit and expand your code with confidence and helps you identify bugs as you write or modify your programs. In conjunction with the Agitar Management Dashboard, Agitator delivers unprecedented productivity, visibility, and control over the developer testing process, leading to profound software quality improvements and cost reductions.

Learn more about Software Agitation, and visit [www.agitar.com](http://www.agitar.com) today!



# Does Your Project Need a Rule Engine

by Chris Moran

## Separating the business rules from the application logic

Many Java developers today have moved toward some form of logging and/or unit test framework, and their code has been purged of many `System.out.println()` statements that were the traditional approach. Now perhaps it's time to get rid of some of those `if (x) { . . . }` as well. Nothing in a piece of code seems to foul up the design as much as the business logic, and going from two-tier to three-tier to *n*-tier hasn't done much to solve that problem.

In some dark layer of the code there is still a tangled mass of ifs, elses, and look-up tables upon which the whole thing rests. After a project goes through a few generations of developers, the "business rule" layer becomes a Gordian knot that defies both change and comprehension. Rule engines offer a framework for isolating the business logic in your applications; this framework is simpler and more flexible than look-up tables. As a design approach, it fits Java programs of almost every conceivable form, purpose, and budget. Most important, it brings the elusive goal of code reuse a step closer. This article lays the foundation for getting started using a rule engine on your own project.

### What Is a Rule Engine?

A rule engine is a system for applying some set of if/then rules to the data in a system and then taking some action on the data or the system itself. Its primary purpose is to separate the business logic from the system logic – to externalize the business logic so it can be maintained separately. Use a rule engine to separate the if/then rules from system code – to externalize them so they can be maintained separately from the system code. The behavior of the system can then be modified without changing code or needing to recompile/redeploy. Rules are stored in human-readable form in a file so they can be changed with a text editor or rule editor.

For example, a typical storefront system might involve code to calculate a discount:

```
if (product.quantity > 100) {
    product.discount = 2;
} else if (product.quantity >= 500 && prod-
```

```
uct.quantity < 2000) {
    product.discount = 5;
} else if (product.quantity >= 2000) {
    product.discount = 10;
}
```

A rule engine replaces the above with code that looks like this:

```
ruleEngine.applyRules(product);
```

The code no longer contains any reference to quantity or discounts. That logic is being handled entirely by the rule engine. This example is something of a straw man since this kind of rule is usually handled with name-value pairs in a database. However, it illustrates the main goal of separating system behavior from system code.

### Do You Need a Rule Engine?

Rule engines have traditionally been used in financial applications for such things as credit scoring and underwriting because of the many and complex business rules these applications require. Coding such rules directly into the application makes application maintenance difficult and expensive because they change so often. It also makes even the initial release of the application difficult because such rules are often changed

between the time the code is written and the time it's deployed. So rule engines were devised to separate the business rules from the application logic.

However, virtually every application has some program logic that needs to change often or needs to change in ways not anticipated in the original design. The real question then is not do you need a rule engine but how much time and money will the rule engine save you? Even if you have only a small collection of rules that are subject to change, your project can benefit greatly by separating them from the rest of the program logic. This is particularly true during user acceptance testing when missed requirements and incorrect assumptions become evident. A rule engine enables you to make dramatic changes in system behavior without dramatic changes in your code, and it enables you to make changes at runtime.

A rule engine is also easier to use and integrate than a database table. If your code employs if/then logic based on look-up tables, it can be greatly simplified with a rule engine.

Quantity	% discount
100	.02
500	.05
2000	.1

Table 1 Typical database lookup table



Chris Moran is a senior Java developer and the chief architect of rules4j. He lives and works in the Washington, DC, area.

cmoran@rules4j.com



Figure 1 Classes for the installer application

Why is a rule engine better than name-value pairs in a database? Name-value pairs are a tried and true way of handling business rules in a system. If we have a system that needs to compute a product discount on the basis of quantity purchased, we might create a table in a database that stores percent discounts by quantity (see Table 1).

In our code we would construct a layer to manage querying this table with something like `select discount from discount_tbl where quantity = 100`. Then in the code that handles business logic, we would plug in a variable to handle the value returned by this query. If the discount ever changes, we simply modify the contents of the table. This works fine until the time comes when we want to stop using quantity to calculate discount and use some metric for customer buying patterns instead. What if we only want to do this for a single product for a few days? Basically we can't. We would have to recode the system to query by buying pattern, add new tables to the database to store the pairs, and create new program logic. Then we would have to undo it all a few days later.

Why does this well-used design turn out to be so inflexible? Because most of

the rule is still coded into the system. Both the attribute being checked as well as the action being taken – the behavior of the system – are hard-coded. We have flexibility only in the degree to which something is done, not in what is being done.

By contrast, a rule engine provides the flexibility to change not only how much, but what, when, where, or any other basis you can imagine. There is no query, no table, and no rule-specific code. There is only a call to the rule engine passing raw data and getting back processed data. All of the logic of what to change as well as the basis for that change is controlled by the rule engine. Before, we were limited to the tweaking of values; now we can change the way the system lets us do business.

### The Basic Flow

How does a rule engine fit into the flow of an application? Basically, it's just another class that you instantiate or to which you get a remote reference. Data goes in, rules are executed, and data comes out. You can also write rules that perform some action on the system based on the data you pass it. When the data is passed in, the rule engine interrogates the object to see which conditions

in your rules are satisfied. If, for example, you have a rule:

```
If quantity > 100 Then do something
```

the rule engine will call the `getQuantity()` method on the object being passed in and compare the return from that call to the value "100." Then it will do something depending on whether the comparison is true or false.

A rule engine sits in your application in exactly the same way as a specialized class written to handle business rules.

### Rule Engine Integration

Most of the rule engines available today loosely implement the JSR 94 specification, providing some commonality to integrating engines from different vendors. Thanks to JSR 94, integrating a rule engine into an application requires very few lines of code. It can be summed up in these few general steps:

- Obtain an instance of the RuleRuntime object.
- Get a RuleSession object from the RuleRuntime.
- Pass data to the rule session object.
- Execute the rules.

 <h1 style="color: red;">Schmo.</h1>	 <h1 style="color: white;">Pro.</h1>
<p><b>Gets job advice from the newspaper.</b></p>	<p><b>Goes to Dice.com first for complete tech career guidance.</b></p>
<p><b>Dice™</b> Look to the tech leader first.™</p>	
<p><b>Only tech jobs • Career guidance • Salary info • Daily job notification • Training &amp; certification</b></p>	

© 2004 Dice Inc.

In a typical application, the first two steps occur once. Getting RuleSessions, passing data, and executing rules happen as many times as needed.

Listing 1 shows how it happens in the code for a typical Web-based insurance application using the rules4J rule engine.

That's really all there is to it. Lines 3, 8, 13, and 15 in Listing 1 carry out each of the four general steps outlined above.

Line 4 obtains an instance of the Rule-runtime object. The two parameters passed here identify the path and name of the XML file that describes this particular rule server.

Line 9 gets a RuleSession object from the Ruleruntime. A rule session handles the execution of a single ruleset. The name of the ruleset in this example is seen in the line that sets the string `m_currentRuleSetName`:

```
TestServer:Test:EligibilityRuleset
```

The three components of this name are `servername : rulebase : ruleset`.

Line 14 adds a List of objects to the RuleSession. The basic operation of the RuleSession is to fire the ruleset once for each object in this List and apply the rules separately to each object. In our example, we put only one object into the List, but there is no limit.

Line 16 instructs the RuleSession to execute the rules. The call to `getObjects()` on line 17 is not really needed because the RuleSession is not a remote object and it is acting on references to the objects loaded into it. The JSR 94 specification actually calls for `StatefulRuleSession.executeRules()` to return a void. Only `StatelessRuleSession.executeRules()` returns a List.

### A Real-World Example

This real-world example comes from an installer program I am currently working on. It creates a custom build and install of a system into a J2EE container. The motivation to use a rule engine for this problem is that the behavior of the installer can vary considerably depending on the OS, container, port assignments, and paths a user selects for the installation. This behavior has the potential to change frequently either because container specifics change from version to version or there's some new system configuration we want the installer to handle. It also seems a waste to write a new and different installer program for the demo, eval, and production versions of the system being installed. Instead, we would like to have one general purpose installer and be able to control the behavior through rules.

What do we put into the rule engine and what do we leave in the program? A

general rule of thumb is to put system capability into your code and system behavior into your rules. For example, the code will have the capability to compile a Java file and create a JAR file with it, but the rules will describe the behavior that causes it to happen. The rules dictate which files get compiled and placed into a JAR, and the code does the actual JARing and compiling. If there is logic that's specific to the mechanics of compiling or creating a JAR file, it should probably stay in the code.

This rule of thumb helps to shape the purpose of each of the classes the installer will need. The classes for the installer application are shown in Figure 1.

- **Installer:** Handles presentation of the GUI.
- **OSCommand:** Handles the many OS commands the installer will need to be able to manage (e.g., copying files, creating directories, JARing files, compiling, etc.). It contains all the logic that pertains to the mechanics (or capability) of the application.
- **InstallParams:** Models data collected from the user – container type, install locations, port assignment, etc.
- **InstallProcess:** Drives the actual install process (taking input from the InstallParams class) and invokes the rule engine. It encapsulates the behavior of the application.

We are really interested in what happens inside the `install()` method of the `InstallProcess` class. This method is where we find the main logic that must sort through the user inputs and decide how to configure and install the system on the user's machine. User input is stored in the `params` object and passed from the `Installer` class. Listing 2 shows the logic as it might have been written in plain Java. (Listings 2-4 can be downloaded from [www.sys-con.com/java/sourcec.cfm](http://www.sys-con.com/java/sourcec.cfm).)

Clearly, such code will need to be changed often as we add capability and respond to vendor changes. In Listing 3 we've taken all this logic and expressed it in an XML file. Different rule engines implement the expression of rules in very different ways, and JSR 94 has nothing to say about how the rules are implemented. Some vendors do this with XML and some have developed specialized rule languages to handle the task (e.g., Fair Isaac's Blaze). The XML file in Listing 3 follows the implementation for the rules4J rule engine.

With the rules now expressed in an external XML file, the `install()` method can be rewritten to invoke the rule engine. Since we initialize the `Ruleserver` in a constructor, the code in the `install()` method is reduced to just a few lines (see Listing 4).

The installer has been improved in two important ways:

1. The `install` method is now very general and, in fact, the `InstallProcess` class has become reusable. Because the rule engine doesn't care what kind of object is being passed to it, we no longer need to write the `install()` method to take an instance of any particular class. This leaves us free to pass all manner of classes that contain user input. The only stipulation is that the rules we write in the rule engine match the properties in the parameter class.
2. Almost anyone can read, understand, and modify the behavior of the class, and they can do so without needing to recompile the code. This is a tremendous advantage to test teams, customer support, and even the developers who must maintain it.

### A Look at the Rule Engines Out There

There are a great many rule engines to choose from. Below is a sampling of the products out there.

#### *ILOG JRules and Fair Isaac Blaze*

JRules and Blaze have been the main players in the business rules rule engine market for some years. They have a well-established user base in the banking and insurance industries. These products are particularly well suited to enterprise applications because they offer a variety of tools for nonprogrammers and administrators. The only drawbacks to either of them are the price and the learning curve. If you are building a main-line business application in the financial services or insurance sectors, they are the obvious choice. [www.ilog.com/products/jrules](http://www.ilog.com/products/jrules) and [www.blazesoft.com](http://www.blazesoft.com)

#### *WebLogic Personalization Server*

BEA has taken the lead in developing JSR-94. Oddly enough, unless you're heavily plugged into the BEA line of products, you may not even realize that it offers a rule engine. In fact, it seems that many on the BEA sales team don't realize it either. I once sat in on a sales demo of WebLogic Portal and Personalization Server and never heard a word about it. Nonetheless, BEA does bundle a rule engine with its Portal product. <http://e-docs.bea.com/wlcs/docs20/p13ndev>

#### *Rules4J and Java Expert System Shell (JESS)*

These products are the emergent players in the field. They are easy to use and inexpensive but sacrifice nothing in basic capability to the bigger players. They are a good fit for most solutions, especially for small projects. They have a short learning curve.



[www.rules4j.com](http://www.rules4j.com) and <http://herzberg.ca.sandia.gov/jess>

### Struts

Didn't expect to see this one here did you? Struts is not a rule engine, but it does have a set of <logic> tags that enable it to evaluate variables and control what is displayed on a JSP page. I've included it in the list only because I've been asked how it compares with rule engines on numerous occasions. If you're building a Web application with Struts, be sure to look at this capability. <http://jakarta.apache.org/struts>

### Conclusion

Rule engines have been around for a long time. Charles Forgy designed the Rete (Latin for net) algorithm employed by most implementations back in 1982. Unfortunately, price and lack of uniformity have kept them out of the hands of developers on most projects. This has changed of late. Inexpensive, powerful, standards-based rule engines are available for projects of almost any budget – a good turn of events for systems engineering. The separation of system behavior from system capability is a key requirement for code reusability. The

failure to separate behavior and capability has an inestimable cost, especially if Java's 2.5 million developers fail at it consistently. Hard-coding behavior certainly keeps us busy, but is it really a productive practice? ☹

### Related Reading

- *JSR 94 Specification*: [www.jcp.org/en/jsr/detail?id=094](http://www.jcp.org/en/jsr/detail?id=094)
- *Some good explanations of the Rete algorithm*: <http://herzberg.ca.sandia.gov/jess/docs/52/rete.html> and [www.cis.temple.edu/~ingargio/cis587/readings/rete.html](http://www.cis.temple.edu/~ingargio/cis587/readings/rete.html)

#### Listing 1

```
1 public void init(HttpServletRequest request)
2 {
3     // Initialize the ruleserver
4     m_ruleRuntime = new
Rules4JServer(System.getProperty("DataDirectory"),
               "rules4JServer.xml");
5
6     person = getPersonData();
7     m_currentRuleSetName = "TestServer:Test:EligibilityRuleset";
8     // Get a rulesession object from the ruleserver
9     m_currentRuleSession = (StatefulRuleSession)
m_ruleServer.createRuleSession
(m_currentRuleSetName, null,
RuleRuntime.STATEFUL_SESSION_TYPE);
10
11     List personList = new ArrayList();
```

```
12     personList.add(person);
13     // Pass the person object to the rule engine
14     m_currentRuleSession.addObject(person);
15     // execute the rules in the rulesession
16     m_currentRuleSession.executeRules();
17     personList = m_currentRuleSession.getObjects();
18     m_currentRuleSession.reset();
19
20 }
21
22 private RuleRuntime m_ruleRuntime = null;
23 private Collection m_ruleSetNames = null;
24 private String m_currentRuleSetName = "";
25 private com.rules4j.RuleSet m_currentRuleSet = null;
26 private com.rules4j.StatefulRuleSession m_currentRuleSession = null;
```

# WebCharts3D

## One demo worth your download

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



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

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

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

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



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

# StAX: Java's XML Pull Parser Specification

An overview

by David Stephenson

Until recently Java programmers have had three options when wishing to access the XML infoset: they could use DOM, JDOM, or SAX. With the release of the JSR 173 StAX specification, Java programmers now have a fourth option, which gives them the efficiency of SAX with a convenient and extendable programming model. This article explores the rationale behind StAX's pull parsing model and describes how you can use the API to more cleanly create Java code to extract the information you need from your XML document. The article describes StAX's two API flavors, "cursor" and "event," and provides some of the reasons why the specification ended up containing two sets of reading and writing APIs. Example usage of each of the reading APIs and each of the writing APIs will be provided.

## Document Streaming vs Document Object Model

When creating code that processes XML documents there are two approaches to dealing with the XML infoset data: object model and streaming. With object model, you first create an in-memory object model tree that holds the complete infoset state for an XML document; once in memory you can freely navigate around the tree and even evaluate arbitrary XPath expressions against the tree. This flexibility comes at a price – the complete details of the XML document must be held in memory as objects for the entire duration of the document processing. The creation of the document object graph requires considerable processor resources and takes up a large memory footprint. This may not be a problem for small XML documents, but for large XML ones memory may become a bottleneck to application performance.

With the streaming approach the XML infoset is processed in a serial manner (as a depth first traversal of the XML infoset tree); once an element has been seen its state is discarded and may be garbage collected. Only the infoset state at the current point of the document is available at any one time, which clearly limits the types of processing that can be done

and implies that you need to know what processing you are going to perform before reading in the XML document. If you don't think you will ever need to use XML streaming, try loading a 100 megabyte XML file into your latest application and watch what happens.

## Streaming Pull Parsing vs Streaming Push Parsing

If you are creating an application that's memory limited, either because you are running on a constrained device (as in a phone) or your application is simultaneously processing several requests (as in an application server), you need to read your XML documents using a streaming model. In the past you were restricted to using the Simple API for XML (SAX). SAX was the first widely available API for reading XML in Java and provides a very low-level, efficient API that deals directly with the character data in the XML document. SAX uses a push processing model in which the SAX library reads the XML document and calls methods on your application objects as it encounters elements and text within the XML document. Although the SAX API is simple, the code application developers need to create to use SAX is not.

A "pull" parsing alternative to SAX's "push" parsing has lurked in the background for some time, but no longer: the recently ratified StAX specification now standardizes a pull parser for Java. StAX provides an alternative processing model where you call methods on the parser at your leisure and move the processing along at your command. The key difference here is that with SAX you don't have control of the application thread and can only accept invocations from the parser. In contrast, with pull parsing you own the application thread and you control when and where you call the XML parser.

This control over "when and where" leads to increased freedoms in application design, allowing you to either collect all your parsing code together or alternatively place your parsing code within the objects that understand that particular type of information.

Pull parsing has the following advantages over push parsing:

- Parsing simple documents can now be done with simple code.
- It's much simpler to write recursive descent parsing code for more complex documents.
- More than one document can be read by an application at one time with just a single thread. This can be useful when part way through reading one document you need to read a second document.



David Stephenson has worked in the computing industry for over 10 years in distributed systems, middleware, and IT solutions. He has a wide range of experience in computing having worked for Hewlett Packard laboratories and for HP's middleware divisions creating e-services technology. Lately David has worked in the area of Web services and on both the Java and .NET platforms and has been HP's contributing expert in the JCP for both JSR 31 and JSR 173.

david.stephenson@hp.com

- The parser can be told to skip parts of the XML document that are not relevant to the application, which simplifies your code and may reduce processing time and memory churn.
- You can create streaming pipelines in an object-oriented way that are efficient and simple to use.

### XML Information Model

The StAX specification models an XML document as a set of events, and these events are pulled by the application and supplied in the order in which they are encountered in the XML document. The StAX specification defines the following types of events: Attribute, Characters, Comment, StartDocument, EndDocument, StartElement, EndElement, Namespace, DTD, EntityDeclaration, EntityReference, NotationDeclaration, and ProcessingInstruction.

The last five event types are only seen if your document contains a DTD. Each event has different properties associated with it depending on the type of event.

A StAX parser is an engine that reads a Unicode character stream and converts the textual data into events. Below is an example XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<pre1:foo xmlns:pre1="http://www.example.com/foons">
  <pre1:sometext>
    the text <![CDATA[<notallowed> as normal text]]>
other text
  </pre1:sometext>
  <pre2:bar ratio="5.5" xmlns:pre2="http://www.example.com/barns"/>
</pre1:foo>
```

The above document would be parsed into the events shown in Figure 1.

As you can see, this small document would be converted (by default) into a stream of 14 primary events. Each colored circle in the figure is an event object. The large circles are the primary events that are seen by the application, while the small circles are secondary events that are generally accessed from some primary event.

Salient points about the event stream to note are:

- Every StartElement event has a matching EndElement event, even for empty events like <eg/>.
- Attributes, although events, are not (normally) seen in the event stream but are instead accessible from their StartElement event.
- Namespaces events are not (normally) seen in the event stream but instead appear twice, first accessible from a StartElement and, second, accessible from the corresponding EndElement.
- XML Character data may be split over more than one event and crop up where you might not expect.

While parsing an XML document the StAX parsing engine maintains a namespace stack. The namespace stack holds details of all the XML namespaces defined for the current element and its ancestors. This namespace stack is accessible through the interface javax.xml.namespace.NamespaceContext, which includes methods for looking up a namespace URI given a prefix and looking up a prefix given a namespace URI.

### Cursor vs Iterator

#### The Story of Two APIs

When programming in Java, object creation has traditionally been seen as the enemy of performance. One of the SAX API's main advantages is that very few superfluous objects are created during the parsing of an XML document. SAX even gives the application direct access into the parser's internal character buffer in order to read text content! This lean and mean approach leads to very efficient parsing of XML. One of the design goals for StAX was for it to be at least as fast as SAX.

Very early on during the process of creating the StAX specification two alternative API styles were proposed by the expert group. One, which I'll call "cursor," followed SAX's lean and mean approach; the other, which I'll call "iterator," was a modern object-oriented API utilizing immutable objects. The expert group looked long and hard at which API style we should go with, including doing a performance analysis of the two styles. What we dis-

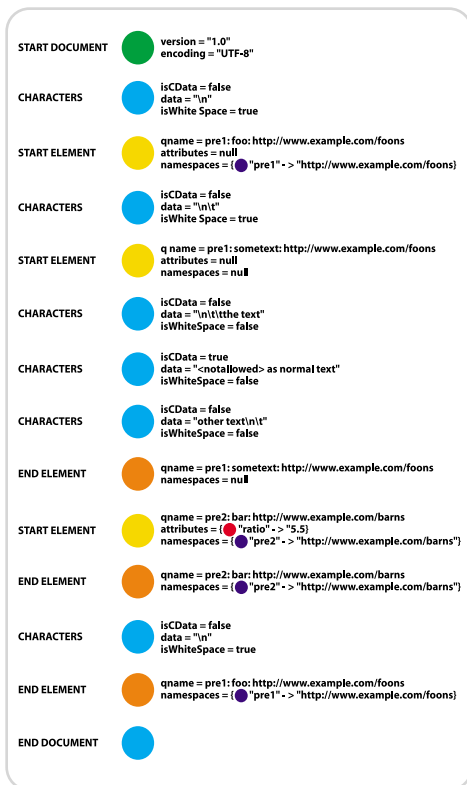


Figure 1 An Event stream

# Why Settle For "Sorta Close?"



## Get The Workflow That Fits

*Reactor 5 – the ideal solution for Workflow Automation, Business Process Integration and Web Services Orchestration*

### Fits Within Your Architecture

J2EE-based, XML-driven, platform neutral

### Fits Your Business Requirements

The extensibility your developers want, the simplicity your business users demand

### Fits How You Want To Buy

Flexibly priced, with source code access

Download your free evaluation copy at [www.oakgrovesystems.com](http://www.oakgrovesystems.com), or contact us at 1.818.440.1234. We can't wait to help you...

**Declare Your Workflow Independence!™**



© 2004 Oak Grove Systems. All rights reserved. All product names are trademarks or registered trademarks of their respective companies.

covered was that we were trying to support three different end-user developers:

1. **Library and infrastructure developers:** The group who creates app servers, JAXM, JAXB, JAXRpc, implementations, etc., and needs a very low-level and close-to-the-metal API with little overhead and few requirements for extensibility.
2. **J2ME developers:** This group wants an XML pull parsing library that's small and simple and has a tiny footprint. They have little need for being able to extend the parser or modify the event stream.
3. **J2EE and J2SE developers:** This large group generally wants a simple, efficient pull parser that naturally produces elegant bug-free code while allowing for more complex features such as stream modification and introducing new application event types.

We looked at many inventive ideas on how to create a single API that would allow us to “have our cake and eat it,” but each of these ideas was rejected as they left us with a bad taste in our mouth and invariably produced a less-predictable API. In the end our desire to support these three developer types and our requirement that we be just as fast as SAX led the expert group to support both API styles.

*Cursor API*

The cursor API contains a central parser interface called XMLStreamReader that includes accessor methods for all the possible information you could retrieve from the XML Information model. The parser interface contains methods for accessing the document encoding, element names, attributes, namespaces, text content, processing instructions, etc. Methods are provided to allow access into the internal character buffer just as in SAX. The cursor approach is like a mirror image of SAX and provides direct access to string and character information while exposing methods with integer indexes for accessing attribute and namespace information just as in SAX. Thus it's possible to access all of the Information Model via a set of methods that return strings so object allocation is kept to a bare minimum.

Listing 1 provides some example code that uses an XMLStreamReader instance called “sr”, which reads the example document listed in Figure 1. The code uses “sr” to walk over the document and retrieve the text content of the element <pre1:sometext> and the value of the ratio attribute of the element <pre2:bar>.

Listing 1 assumes the XMLStreamReader sr has just been created, i.e., StartDocument will be the first event.

*Iterator API*

The iterator API presents the event stream as an ordered list of immutable event objects. The StAX API defines a common base interface called XMLEvent and a subinterface for each of the event types listed in the XML information model. The Iterator API

contains a central parser interface called XMLEventReader with just five methods in it, the most important being nextEvent(), which returns the next event in the stream. The interface XMLEventReader implements java.util.Iterator so it can be passed into routines that can handle the standard Java Iterator.

The common super interface XMLEvent contains methods for finding the actual event type and downcasting to the event subtypes. Listing 2 shows the equivalent code for reading our example XML document but using the XMLEventReader “er”.

Clearly in a real application the three QName objects would be held in static final fields but are left inline here to aid in comparing the code examples.

What can I do with the iterator API that I can't do with the cursor API?

As the XMLEvent subclasses are immutable objects, you can place them in arrays, lists, and maps and pass them through your application as you desire, even after the parser has moved on to later events.

You can create your own subtypes of XMLEvent that are either completely new information items or extensions of existing events but with extra methods.

You can modify an event stream by adding or removing events in a way that's much simpler to code than with the cursor API.

**Which API Should I Use?**

This decision can only be made by the individual developer depending on the specific situation; if one API fitted all needs we would not have ended up with two.

My personal rules of thumb:

1. If you're programming on J2ME, use the cursor API.
2. If you're creating low-level infrastructure or libraries and you need to achieve the best possible performance, use the cursor API.
3. If you want to create pipelines of XML processing, use the iterator API.
4. If you want to modify the event stream, use the iterator API.
5. If you want to future proof your app by enabling plug-gable processing of the event stream, use the iterator API.
6. If in doubt, use the iterator API.

**Performance: What's the Beef?**

During the expert group design discussions for StAX, the author created a prototype pull parser that provided both types of API styles, and a series of performance tests were performed using a wide range of XML test data. Due to the fact that the internal parser implementation was common for all the tests, the results showed the performance impact of using the iterator API style versus the cursor API style when accessing the parser.

The performance differences between the API styles arise mostly because of the extra objects that are created and later need to be garbage collected. The cursor and event API need to create the objects shown in Figure 2.

The cursor API does not need to create string objects for XML character data as it provides direct access to the internal character buffer. In addition, the iterator API needs to create the immutable event objects. The items colored yellow are string objects that may be cached to improve performance at the expense of some cache management complexity.

Figure 3 shows the number of bytes of XML processed per millisecond averaged over different sets of XML test files. The test was run on JDK 122 and JDK 142 for both API styles and with and without string caching enabled. The test files are loaded into memory so there is no I/O during the test runs.

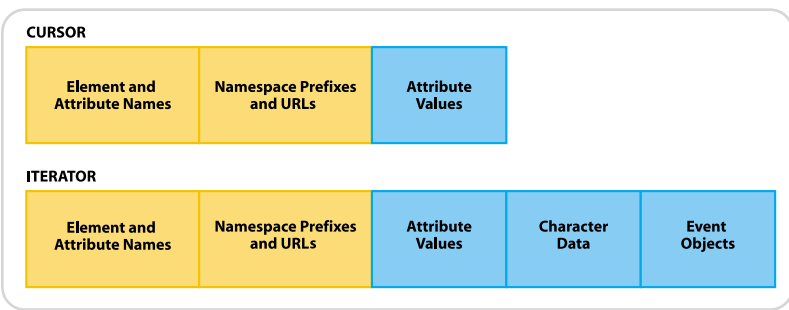


Figure 2 Objects created during parsing



Shrink my **development** time.  
Give me the technology to **deliver** it  
**Faster. Better. Easier.**



**Faster. Outsmart your development deadlines with AMD64 technology.**

**Better. Direct Connect Architecture lets you do more.**

**Easier. Your platform choice is simpler, since AMD64 technology excels across a wide variety of application workloads.**

**Register at [developer.amd.com](http://developer.amd.com) and enter a drawing for a chance to win an AMD64 system. See official rules for details and eligibility requirements.**

## “When programming in Java, object creation has traditionally been seen as the enemy of performance”

Clearly there has been a massive (3–6 times) performance improvement across the board from JDK 122 to JDK 142. The iterator API without any string caching is 6.5 times faster on JDK 142.

String caching makes a big difference on JDK 122, running up to 50% faster, but only a small difference on JDK 142, showing less than a 5% improvement with perfect caching. Clearly we can now take Joshua Bloch at his word when he says that object pooling of lightweight objects is unnecessary with modern JVMs.

The overhead from using the iterator API style instead of the cursor API is around 25–30%. This sounds like a lot but remember this test program is doing 90% XML parsing and 10% application logic, whereas your typical application would probably be the other way round – 90% app logic and 10% parsing – which would drive the overhead down to about 3%, which is in the noise for most applications.

Of course your mileage may vary depending on your particular usage scenario. With the first generation StAX parser coming out soon, we'll see if this level of performance difference is also reflected in the real StAX parsers.

### StAX Input Factories

How do I create an instance of XMLStreamReader or XMLEventReader?

StAX parsers that implement either of these two APIs are created by the `javax.xml.stream.XMLInputFactory`, which follows the standard factory pattern. When we get JAXP support you'll be able to create instances via the JAXP APIs. Create a new instance of `XMLInputFactory` by calling `newInstance()`; this method will search for a StAX implementation using the standard techniques.

The input factory has a set of configuration parameters that control the features of the parser that will be created by

the factory. Once you have an instance of the factory you can override the default configuration parameter values.

Some of the more interesting configuration parameters are:

- **`javax.xml.stream.isCoalescing`**: Defaults to false but when set to true will request a parser that coalesces all contiguous text into a single character event.
- **`javax.xml.stream.supportDTD`**: Defaults to true, but when set to false will request a parser that does not support DTDs in XML documents.
- **`javax.xml.stream.resolver`**: Can be used to set an implementation of `XMLResolver`, which is used during parsing to resolve external entities.

Below is the code to create an instance of the XMLStreamReader on a File f.

```
XMLInputFactory factory = XMLInputFactory.newInstance();
XMLStreamReader sr = factory.createXMLStreamReader(new
FileInputStream(f));
And to create an XMLEventReader
XMLInputFactory factory = XMLInputFactory.newInstance();
XMLEventReader sr = factory.createXMLEventReader(new
FileInputStream(f));
```

If we wanted a differently featured parser, we would set some configuration parameters before creating the parser as follows:

```
XMLInputFactory factory = XMLInputFactory.newInstance();
factory.setProperty("javax.xml.stream.isCoalescing", Boolean.TRUE);
factory.setProperty("javax.xml.stream.supportDTD", Boolean.FALSE);
XMLEventReader sr = factory.createXMLEventReader(new
FileInputStream(f));
```

Some of the standard configuration parameters are optional, meaning that some implementations may choose not to support the feature. You can check to see if a standard (or non-standard) configuration parameter is supported by calling `isPropertySupported()` on the factory instance. Here's an example of using an optional feature:

```
XMLInputFactory factory = XMLInputFactory.newInstance();
if(factory.isPropertySupported("javax.xml.stream.isValidating")){
factory.setProperty("javax.xml.stream.isValidating", Boolean.TRUE);
factory.setProperty("javax.xml.stream.reporter", this);
}
XMLStreamReader sr = factory.createXMLStreamReader(new
FileInputStream(f));
```

The above code instantiates a DTD validating parser if the implementation supports DTD validation.

Once you have created either an `XMLStreamReader` or an `XMLEventReader` parser you can find out what its configuration is by calling `getProperty()` on the parser.

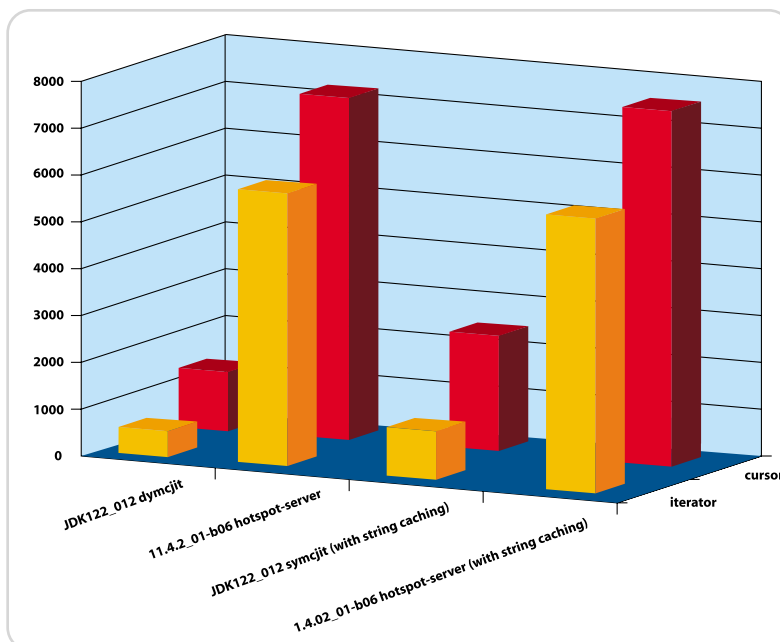


Figure 3 Relative performance of cursor and event API styles

# Has your Integration Project hit an Iceberg?

**Software Licenses**

**9%**

**\$250K**

**Customization & Implementation**

**43%**

**\$1.2M**

**Hardware**

**26%**

**\$722K**

**IT Personnel**

**14%**

**\$389K**

**Maintenance**

**7%**

**\$194K**

**Training**

**1%**

**\$28K**

**\$2.5M  
Hidden  
Costs**

**Cut your costs by over 80% with Fiorano ESB™**

Download free evaluation today: [www.fiorano.com/downloads](http://www.fiorano.com/downloads)

## **Fiorano Integration Solutions**

- > Fiorano Business Integration Suite™
- > Fiorano ESB™, Enterprise Service Bus
- > FioranoMQ™, Java Messaging Server

**Fiorano**  
Enabling change at the speed of thought

One important design feature of StAX is that you must specify the properties of the parser before you create the parser and you cannot change the property value for an existing parser nor set a new data source into the parser. The rationale behind these restrictions is that we wanted to enable optimized and modular implementations.

A StAX implementation could use a special parsing engine with its own optimized byte to Unicode decoding when reading from a `java.io.InputStream`; alternatively it could use a different parsing engine when a `java.io.Reader` is the data source. Another example is the `javax.xml.stream.supportDTD` property that when “false” could trigger an implementation to use a simpler, faster parsing engine. If we allowed any of the configuration parameters to be modifiable after the parser was created, these types of optimization would be considerably harder to implement.

### What About Writing XML?

StAX is a truly bidirectional API and can be effectively used to generate XML, either from scratch or as the result of a StAX Event pipeline. The StAX writers are intelligent in that they maintain namespace stacks and can automatically generate namespace prefixes (if you don't care what they look like). The writers can also close elements using the correct prefix and localname.

Listings 3 and 4 show some code to generate our example XML file using someText and ratio variables. If you want to use the cursor API, the code appears as in Listing 3. If you want to use the iterator API, the code should look like Listing 4.

The writers automatically escape any illegal Unicode characters such as < or & that are found in character content or attribute values.

### Summary

After a lot of hard work the StAX API specification has finally seen the light of day and Java application programmers now have a standard pull parser interface for XML. The StAX API has many advantages over the SAX API for developers, including a simpler programming model and the ability to modify the event stream data and extend the information model to allow the introduction of application-specific additions. J2ME programmers now have an XML API that matches their resource-constrained environments, while library developers now have a standard API to use that fits in with their client applications' threading models and performance expectations.

### Acknowledgment

I would like to thank the members of the JSR 173 expert group for an interesting and stimulating set of discussions over the past two years. ☺

#### Listing 1: Reading with the XMLStreamReader

```
while (sr.hasNext()) {
    sr.next();
    if (sr.getEventType() == XMLStreamConstants.START_ELEMENT) {
        if (sr.getLocalName().equals("sometext") &&
            sr.getNamespaceURI().equals("http://www.example.com/foons")) {
            // get all the text content of <sometext> and
            // store in someText variable
            someText = sr.getElementText();
        } else if (sr.getLocalName().equals("bar")
            &&
            sr.getNamespaceURI().equals("http://www.example.com/barns")) {
            // get the value of the ratio attribute
            ratio = sr.getAttributeValue(null, "ratio");
        }
    } else {
        // ignore event
    }
}
```

#### Listing 2: Reading with the XMLEventReader

```
while (er.hasNext()) {
    XMLEvent event = er.nextEvent();
    if (event.getEventType() ==
        XMLStreamConstants.START_ELEMENT) {
        StartElement se = event.asStartElement();
        if (se.getName().equals(new QName("http://www.example.com/foons", "sometext"))) {
            someText = se.getElementText();
        } else if (se.getName().equals(new
            QName("http://www.example.com/barns", "bar"))) {
            Attribute ratioAttr =
                se.getAttributeByName(new QName("ratio"));
            if (ratioAttr != null) {
                ratio = ratioAttr.getValue();
            }
        } else {
            // ignore event
        }
    }
}
```

#### Listing 3: Writing with the XMLStreamWriter

```
XMLOutputFactory factory = XMLOutputFactory.newInstance();
factory.setProperty("javax.xml.stream.isPrefixDefaulting",
```

```
Boolean.TRUE);

XMLStreamWriter sw = factory.createXMLStreamWriter(new
    FileOutputStream(destination), "UTF-8");
sw.writeStartDocument();
sw.writeStartElement("http://www.example.com/foons",
    "foo");
sw.writeStartElement("http://www.example.com/foons", "some-
    text");
sw.writeCharacters(someText);
sw.writeEndElement();
sw.writeStartElement("http://www.example.com/barns", "bar");
sw.writeAttribute("ratio", ratio);
sw.writeEndElement();
sw.writeEndDocument();
sw.close();
```

#### Listing 4: Writing with the XMLEventWriter

```
XMLOutputFactory factory = XMLOutputFactory.newInstance();
XMLEventFactory eventFac = XMLEventFactory.newInstance();

factory.setProperty("javax.xml.stream.isPrefixDefaulting", Boolean.TR
    UE);

XMLEventWriter ew = factory.createXMLEventWriter(new
    FileOutputStream(destination), "UTF-8");
ew.add(eventFac.createStartDocument());
ew.add(eventFac.createStartElement(null, "http://www.exam-
    ple.com/foons", "foo"));
ew.add(eventFac.createStartElement(null, "http://www.exam-
    ple.com/foons", "sometext"));
ew.add(eventFac.createCharacters(someText));

ew.add(eventFac.createEndElement(null, "http://www.example.com/foons"
    , "sometext"));

ew.add(eventFac.createStartElement(null, "http://www.example.com/barns"
    , "bar"));
ew.add(eventFac.createAttribute("ratio", ratio));

ew.add(eventFac.createEndElement(null, "http://www.example.com/barns"
    , "bar"));

ew.add(eventFac.createEndElement(null, "http://www.example.com/foons"
    , "foo"));
ew.add(eventFac.createEndDocument());
ew.close();
```



Java  
J2EE  
WebSphere  
WebLogic  
.NET  
UML  
GYDTSTUT\*



## \*Give Your Developers The Skills To Use Them.

Over 30,000 developers from hundreds of Fortune 1000 companies have come to InferData over the last decade.

They keep coming because their companies know InferData is the best way to stay on top of the newest technologies.

Here's what clients like IBM, Sears, GE and American Express are saying about us:

- "Excellent course!"
- "Off the charts!"
- "Highly recommended!"
- "Lots of hands-on labs!"
- "Outstanding! Wouldn't consider going anywhere else!"

### World-Class Training.

InferData has a full curriculum on the latest technologies, taught with the precision and thoroughness your developers need to keep in top form. A quick look at our course catalog will prove that we know what we're doing.

### Managers and Developers will benefit.

InferData offers more than 130 courses! Everything from "Managing Object-Oriented Projects: An Overview for Busy Managers" to "Mastering J2EE Development with Rational XDE."

### Consulting and Mentoring too.

Our decade of building enterprise systems can offer invaluable insight into your operations. We can help guide your organization through development barriers and move you quicker to your goals.

### TTIDT\*

\*Talk To InferData Today. We'll show you how your organization can move faster and more efficiently, and how the right information can make a real difference in your development efforts.

**JavaOne** We'll see you at JavaOne 2004,  
June 28 – July 1 in San Francisco.



**Training. Consulting. Mentoring.**

888-211-3421 [www.inferdata.com](http://www.inferdata.com)



Joe Winchester

Desktop Java Editor

# Frameworks – Strongest Support or Weakest Link?

**W**hile software frameworks are always created with the best intentions, I believe that many of them fail for the same reason that any other software project does: a lack of clear understanding by the programmers of who their users are and what scenarios they are trying to solve.

Not all frameworks are bad, and it's impossible to write software without using someone else's code. So what makes a good framework?

## The Framework Team

*The Mythical Man-Month* by Frederick P. Brooks (Addison-Wesley) describes an effect he coins "the second system syndrome." This is where developers working on their second project are at their most dangerous – they try to implement every bell or whistle the first system lacks, and they overengineer and design unnecessary features. These are also the kind of people who unfortunately are drawn to writing frameworks. They easily fall into the trap of abstracting behavior to the point of silliness.

I myself have been guilty of this when working on a software project that was made part of "the framework team." We had just finished building a fairly reasonable program for one set of users on a successful proof-of-concept project, and were

ply didn't document or think through the API clearly enough. I have little patience with these and my solution is often to throw out the framework classes and just strip things back to the bare bones. That way everything in the stack is familiar so you're in control of what your program does, and you're not at the behest of someone else's ideas of what it should be doing.

## Harvest or Grow?

A popular line is "frameworks are harvested and not grown." To a certain extent I agree with the philosophy behind this – until you've actually solved a problem with a concrete implementation, it makes little sense to try and preempt it with fancy code that might be wide of the mark. Having solved the problem once, the next time it's encountered the original code can be refactored to create reusable and shareable building blocks. The idea being that over time, a set of useful frameworks will naturally take shape. While this works on a small scale, often for the sole benefit of a single programmer encountering similar problems again and again with a set of sharper tools each time, I believe it doesn't scale into large application frameworks. A good set of class libraries or application frameworks needs to be designed from the ground up. The API must be well thought out

“ The most frustrating type of problem is one encountered because a framework developer was trying to be clever and squelched an exception or simply didn't document or think through the API clearly enough ”

embarking on the entire enterprise-wide application with integrated software from accounting to invoicing to CRM. We had our own room and sat for over a year designing the ultimate set of software frameworks to solve the problem. We all fell horribly into the trap where we believed programmers using our frameworks would just want to lazily cookie cut end-user applications with just a few mouse clicks; we created software tools for them rather than useful class libraries and documentation.

The project failed to deliver anything that ever made it in front of an end user and, scarily, at the next company I worked for, I found a similar doomed group tucked in a corner building with their own ultimate set of frameworks. My conclusion is that if your end users are other programmers, make sure it's them you are always trying to help and don't try to second guess their issues – talk to them, listen to them, and create something that is useful and helpful.

## Die Gracefully

At my current job I spend quite a bit of time debugging other people's code. When you run into trouble it's always good to get another pair of eyes to review your work to spot something you may have missed. The most frustrating type of problem is one encountered because a framework developer was trying to be too clever and squelched an exception or sim-

and thoroughly documented with Javadoc and example code. A framework that grows organically often turns out to be someone else's harvested garden mulch.

## Everyone Wants to Write Frameworks

A great line I heard at a presentation many years ago was "everyone wants to write a framework, but no one wants to use one." The reasons for this are the ones described above: the framework hasn't been thought through properly. It's evolved from someone's set of toolbox classes with no clear API or use cases, and has been developed by a programmer working on his second system and daydreaming about creating a new fourth generation language.

Despite all of this, we need more frameworks. Too much day-to-day programming should be abstracted into application frameworks, class libraries, and the language itself. I'd much rather spend time with end users trying to build software to help them with their business problems than writing some grungy application code that I wish had been available for use in a set of well-documented frameworks. Before anyone begins writing another framework, however, please put yourself in your users' mindset and ask what they want and provide them with it. Don't fall into the trap of providing what you think they want, or even what you yourself want. The two are not the same. ☛

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

joewinchester@sys-con.com

 WebAppCabaret™

<http://www.webappcabaret.com/jdj.jsp>  
1.866.256.7973

J2EE Web Hosting

Quality Web Hosting at a reasonable price...

## <JAVA J2EE HOSTING AND OUTSOURCING>

JAVA Developers: Design and Architect with Struts. Develop with Eclipse. Build with Ant. Deploy to WebAppCabaret. It is that easy with a hosting company that understands the Java J2EE standard. Web Services - a buzz word or reality. How many web hosts provide the facilities to deploy and run applications written for Web Services? We Do. At WebAppCabaret, we lead the way in providing comprehensive Java J2EE and PHP/Perl Web Hosting solutions.

Easy Application Deployment is a reality at WebAppCabaret. Each Account's J2EE Application Server is installed with its own instance and default settings so you have complete control. Our Web Control Panel makes it a breeze for JVM restarts. Such is an example of our Advanced Web Hosting Infrastructure and Tools. If you are a consultant, do you have a complex web hosting requirement for your client? Web Host or Reseller, are you looking to provide services without the headaches of managing your own network infrastructure? Look no more.

For JAVA J2EE we offer the latest versions of Tomcat, JBoss, and Jetty Application Servers. For Scripting programming we offer the latest PHP and Perl. We also offer the latest MySql and PostgreSQL Databases. Commercial software, including Resin, JRun, and Oracle, is also available for an extra charge. In addition we provide valuable tools such as Bugzilla Bug Tracking, CVS Version Control plus a whole lot more. Does your service provider offer ENCRYPTED ACCESS for Email (POP3S/SMTSPS/IMAPS), Shell, and FTP? WE DO. All of this is backed by our Tier 1 Data Center with 100% Network Uptime Guarantee.

Below is a partial price list of our standard hosting plans. (Reseller accounts also available). For more details please log on to <http://www.webappcabaret.com/jdj.jsp> or give us a call at 1(866)-256-7973.

### \$39/mo Enterprise

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

### Options

The following are some of the add-on options with the associated extra starting at costs:

Resin: \$10/mo

300MB Oracle: \$40/mo

JRun: \$20/mo

ColdFusion: \$30/mo

### \$191/mo Dedicated

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

### \$2000/mo 4Balance

4Balance is our entry level High Availability Load Balancing service comprised of:  
1 Database Server  
4GB RAM  
Dual Xeon  
2 Application Servers  
2GB RAM  
Single Xeon  
1 Load Balancer  
1GB RAM  
Single Xeon

### \$99/mo Reseller

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

10 Professional Plans  
Your Web Site  
NGASI Control Panel with your own Logo  
50% Discount  
No System Admin  
Easy Account Mgt  
*more...*

# Never Too Rich or Too Thin

by Bernhard Wagner

## Following the middle road

**Y**ou can never be too rich or too thin. That's what Wally Simpson might have quipped to her stock trading application had she lived to enjoy the blessings of the Internet. Indeed, Wally may have had a point there: today's mainstream approaches to end-user computing are lacking. Fat clients are difficult to distribute and HTML is inadequate for high-end GUIs. A client that is both rich and thin would be the ideal solution.

### Rich Thin Clients

Java is a great platform for rich thin clients (RTCs). The availability of JREs on both the client and server is a formidable basis for an RTC library. Given standard Java infrastructure, such a library can offer server-side peer objects for widgets and a presentation engine executing the GUI for any number of applications, as shown in Figure 1.

This library will be lean and mean because it can:

- Delegate event handling and graphics functions to the JRE on the client
- Profit from the fact that JRE runs both as a plugin within a browser and on the desktop
- Make use of Java Web Start for distribution
- Leverage J2EE for communication and server-side modeling of the user interface



Bernhard Wagner is an independent software consultant and longtime developer of rich-client software. He developed a visual programming environment allowing the visual composition of Multimedia components, 3D direct manipulation applications, and numerous HTML applications.

bw@xmizer.biz

An RTC library is not a panacea. Yet it may go a long way toward rich GUIs without getting chubby or sacrificing the core advantages of HTML. As we'll see later, a number of products testify to this point today.

Let me discuss some key characteristics of a well-designed RTC library and how it can profit from Java.

### Never Too Rich

From the perspective of usability, an RTC presentation engine should support as many rich client functions as possible. The functionality must be

significantly better than HTML's, offering:

- A more responsive interface that minimizes server round-trips
- Direct manipulation
- Comfortable widgets like tables with self-sorting columns, trees, and high-end editors
- Superior integration of desktop functions

### Never Too Thin

From the point of view of distribution and operation, an RTC presentation engine should be as lean as possible, that is:

- Free of application-dependent code so that the rollout of applications is server-side only
- Sufficiently small to enable distribution as an applet
- Use existing J2SE and/or J2ME infrastructure wherever possible

Evidently there is a trade off between the wish lists for thin and rich. This suggests a further requirement: an RTC engine must be as lean as possible in its basic form, but extensible. It should come as a slim core library with plug-gable extensions and an API allowing the integration of existing rich client libraries.

### Optimized Communication

An important bonus of Java-based RTCs is that their network bandwidth

requirements can be minimized. Typically communication will be several times more efficient than for HTML for the following reasons.

First, server round-trips can be slashed by executing tasks within the presentation engine, for example:

- The enabling and disabling of widgets that depend on each other
- Syntactic validations and formatting of text fields
- Sorting of lists or columns
- Caching

A second means to minimize communication is to model the status of user interfaces on the server. This enables the session to keep track of what is visible on the client and thus limit data transfer to visible items.

### Server-Side Programming Model

Designing a client/server application is substantially simplified when a server-side programming model is employed. This avoids the difficult issue of splitting functionality between client and server.

A well-designed RTC library enforces a server-side model. Notice that this will exclude an approach in which developers specify code that is transferred to the client and executed there. This latter approach leads to fat client programming, which is undesirable.

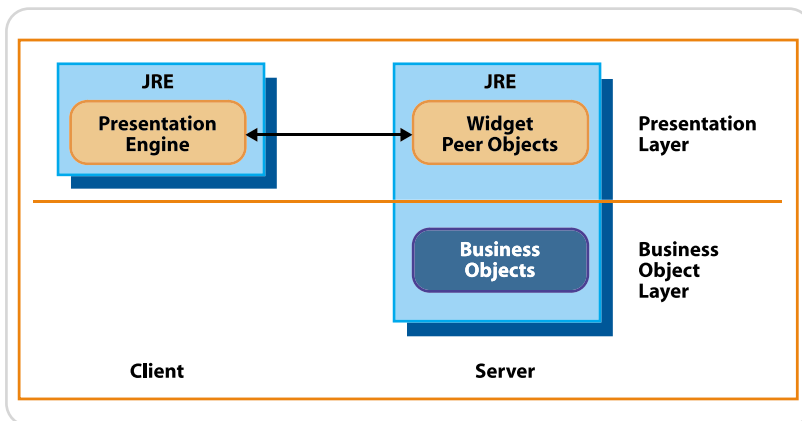


Figure 1 JRE-based architecture for rich thin clients

## Overcoming Unit Testing Obstacles

Unit testing, as commonly performed, is a difficult and resource-intensive process. First, the developer must analyze the code and determine how to add tests that would best verify the unit. Next, the developer must build a test framework. Even when the framework is well-defined (as it is for JUnit test cases), constructing each framework still requires coding. Finally, the developer must design and implement test cases, ensure that the test performs the necessary setup, and execute the test cases.

When a team begins to implement unit testing, the team members are typically excited by the novelty of the practice and they start writing test cases. However, the novelty soon wears off. In fact, unit testing is often abandoned altogether—especially by teams that do not have a way to automatically generate unit test cases, then expand these test cases to verify specific requirements.

Unfortunately, even the best unit testing tools cannot—on their own—ensure that a team performs unit testing. Truly implementing a practice such as unit testing in a team environment involves more than just tools. It also requires the team culture, workflow, and supporting infrastructure needed to embed the practice into the team's development process.

To help development teams make unit testing an enduring part of their development process, Parasoft has defined the unit testing practice as part of the Parasoft AEP Methodology, a comprehensive strategy for preventing errors in a team environment. To learn how this methodology works, visit <http://www.parasoft.com>.

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

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



## Introducing Parasoft Jtest®

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

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

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

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

For Downloads go to [www.parasoft.com/jtest](http://www.parasoft.com/jtest)

Call 888-305-0041 x3303 or email: [jtest@parasoft.com](mailto:jtest@parasoft.com)



### FEATURES

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

### BENEFITS

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

### PLATFORMS

Windows 2000/XP  
Linux  
Solaris

### Contact Info:

Parasoft Corporation  
101 E. Huntington Dr., 2nd Flr.,  
Monrovia, CA 91016  
[www.parasoft.com](http://www.parasoft.com)

## Seamless Object-Oriented API

A further feature for an RTC library is a seamless object-oriented design and API. Ideally, an RTC library offers a server-side API that corresponds to the API of a well-known library like Swing, AWT, or SWT.

The benefit of such a design is that a cumbersome mix of technologies can be avoided: instead of cobbling the GUI together with Java, JSP, HTML, or proprietary XML languages, the developer can use a seamless Java API.

## Server-Side Execution

As mentioned, downloading user-defined code to the client for execution is undesirable. An RTC system must execute everything on the server, except for the GUI.

In fact, even the model of the GUI must reside on the server in order to optimize communication (see above).

Notice that server-side execution is also an advantage for security as it can be handled much easier on the server.

## Pluggable Communication

The communication protocol is one of the major issues for client/server applications. Depending on the environment in which an application must run, different protocols must be used.

For this reason, an RTC library should isolate client/server communication in pluggable modules, such that an application can be configured to run over HTTP, HTTPS, RMI/IIOP, or other protocols.

## Standalone/Offline Execution

With a Java VM on both the client

and server, an RTC library can enable flexible on/offline execution with a single code base.

If pluggable communication is in place, all that's needed is a module that simulates client/server interaction. Such a module will allow you to run a client and server in a single VM and thus on a standalone machine.

Applied cleverly, standalone execution will serve three purposes. It will:

1. Simplify development because the edit/compile/test cycle can be executed locally and within the IDE
2. Support incremental growth – applications starting as a single user can be converted to multiuser with minimal effort
3. Enable development of applications that can run both online and offline

## Leveraging Standards

J2EE, J2SE, and J2ME provide a superb standardized platform for RTC systems. All basic functions required for the GUI, communication, server sessions, and security are readily available. As a consequence, a well-designed RTC library is just a thin software layer that essentially provides a server-side API for the standard widgets of Swing or AWT, delegating everything except the core RTC functions to the standard libraries. An API for SWT is, of course, also an option for cases in which the client-side presentation engine relies on SWT. Figure 2 shows how an RTC library can be embedded into a standard infrastructure.

## Leveraging Existing Infrastructure

Focusing on core RTC functions is a key requirement, not only with respect to standards. An RTC library should be designed to integrate with the existing infrastructure. It should, for example, fit into an existing platform for HTML applications, enabling a mix and match of HTML clients and rich clients, as well as multichannel applications that share everything up to the presentation layer.

An RTC library is, therefore, typically an extension of an existing software platform and not a platform of its own.

## Products on the Market

A number of products illustrate how it can be done: AltioLive, AppProjector, Canoo ULC, Classic Blend, Droplets, RSWT, and Thinlets.

All of these are Java-based RTC libraries. They have put a different emphasis on the defined requirements, and some of them are not pure Java but are hybrid, employing proprietary XML languages. Their common denominator is that they all prove the viability and usefulness of Java for RTCs. Products that forego the advantages of JRE on the client are available as well: examples are Classic Blend and Macromedia Flex. They use JavaScript and a proprietary execution environment on the client, respectively.

## Conclusion

Today's mainstream approaches of the HTML thin client and the fat client productivity application are antipodes. The one's strength is the other's weakness.

The rich thin client (RTC) is the middle road that often succeeds in offering the benefits and avoiding the weaknesses of both. Such magic is not possible for all scenarios, but for many client/server applications.

Various Java-based RTC technologies exist today. Java is particularly suitable for RTC libraries because of its cross-platform availability and broad standard infrastructure. Most important, Java enables a seamlessly object-oriented, server-side programming model that avoids the cumbersome mix of technologies with proprietary XML languages, JSP, HTML, and others.

Now that the rich client is becoming popular again, we may expect that many of those who have experienced the benefits of HTML will not be happy to cope with fat clients again, or with the prospect of building a new infrastructure for client/server computing from scratch. Some of them may go with Ms. Simpson's advice and choose Java RTCs. ☺

## References

- *AltioLive*: [www.altio.com](http://www.altio.com)
- *AppProjector*: [www.asperon.com](http://www.asperon.com)
- *Canoo ULC*: [www.canoo.com/ulc](http://www.canoo.com/ulc)
- *Classic Blend*: [www.appliedreasoning.com/products\\_what\\_is\\_Classic\\_Blend.htm](http://www.appliedreasoning.com/products_what_is_Classic_Blend.htm)
- *Droplets*: [www.droplets.com](http://www.droplets.com)
- *RSWT*: <http://rswt.sourceforge.net>
- *Thinlets*: [www.thinlet.com](http://www.thinlet.com)
- *Classic Blend*: [www.appliedreasoning.com/products\\_what\\_is\\_Classic\\_Blend.htm](http://www.appliedreasoning.com/products_what_is_Classic_Blend.htm)
- *Macromedia Flex*: [www.macromedia.com/software/flex](http://www.macromedia.com/software/flex)

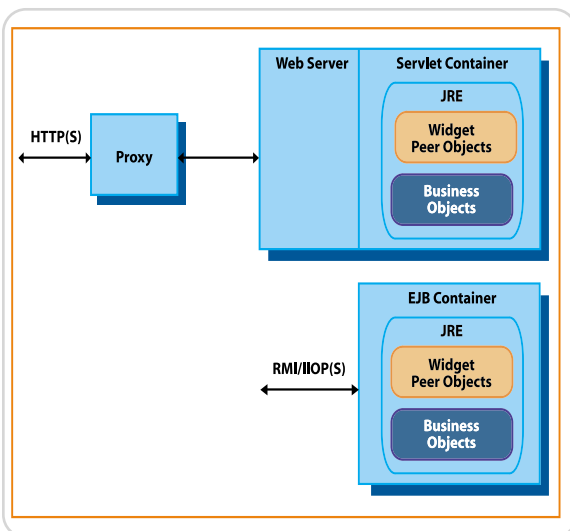


Figure 2 RTC library leveraging J2EE

# DID YOU EVER IMAGINE SUN & MICROSOFT WORKING TOGETHER

?

## SESMA DID

**SESMA** helps software companies create richer and more secure internet web applications by bridging the gaps between Microsoft and Java desktop technologies.

Global 2000 companies have realized that browser-based technologies will not provide sufficient security as they expose their services to the masses over the Net. SESMA SDKs make it possible to:

- Integrate MS/Java solutions using the Java infrastructure
- Capitalize on the security benefits of Java software
- Exploit Microsoft desktop technology where needed
- Encapsulate sensitive processes within the Java Virtual Machine
- Use 3<sup>rd</sup> party technologies to fill in for Java platform visualization

INTEGRATING JAVA INTO WINDOWS<sup>®</sup> IN WAYS YOU'VE ONLY IMAGINED

**THAT'S WHAT WE DO.**

Java Developers command the power of Microsoft desktop technologies using simple, powerful SESMA SDKs.

**Sun and Microsoft as business partners?**

We'll make them work together for you & yours—get **FREE** copies of our Architect<sup>™</sup> software line to help power you along the development curve. Go to :

**[sesma.com/jdj](http://sesma.com/jdj)**

**SESMA**  
**THAT'S WHAT WE DO.™**

# Bringing Mars Down to Earth with Java3D

## EXPLORING JAVA3D

by Michael Jacobs

**Y**ou've probably seen the breathtaking photographs of the surface of Mars from the rovers Spirit and Opportunity. You've also seen the amazing high-altitude photographs taken from satellites orbiting Mars. But let's face it, the rovers must land on flat, boring areas and the satellites are too high above the interesting places like canyons and craters.

*What is an Earth-bound Java programmer to do? Using the Java3D API and publicly available data, you can create pictures like Figure 1.*

The Java3D API is robust enough to handle just about any 3D programming job. The complexity can also make a grown man cry. This article covers the Java3D basics with an emphasis on producing something from another world without the tears. We'll first go over the anatomy of a scene – how to add shapes to the scene, light it, and explore it by moving the view. In the end you'll have a way to explore any location on Mars from any angle moving through the landscape in real time.



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

[jacobs.michael@mayo.edu](mailto:jacobs.michael@mayo.edu)

### What Is Java3D?

Java3D is a sophisticated programming interface intended for rendering three-dimensional scenes and sounds. The latest version is 1.3.1 and supports DirectX and OpenGL. There is substantial support for vector math, animation, lighting, shading, fog, stereo images, collision detection, texturing, loading three-dimensional models from other products, and interactions with the universe you create.

Your Java3D universe consists of a tree structure called a scene graph that contains the shapes, lights, and virtual cameras called views. Java3D automatically renders the scene, properly lighting the shapes based on the view, and changing the view based on user input. Java3D takes care of details like removing hidden surfaces and applying textures to surfaces. The scene graph is the center of your Java3D universe.

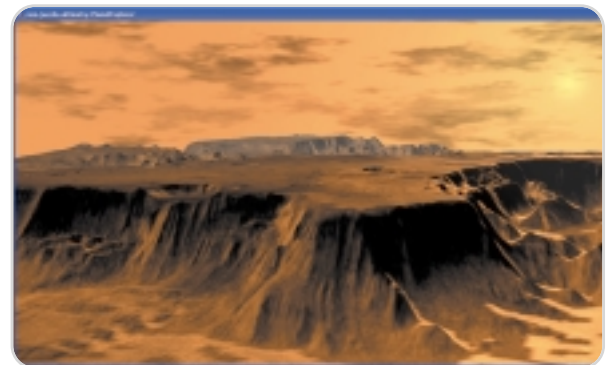
### Creating a Scene

A VirtualUniverse has a scene graph created as a tree structure with a root of a Locale object (see Figure 2). All

other nodes in the scene graph form a parent-child relationship with other nodes in the tree. A scene graph separates the content of the universe from the viewing parameters at the Locale root. The nodes of the content and view branches of the tree are grouped with a BranchGroup. A group node can have multiple children and one parent in the tree. Another example of a group node is a TransformGroup. As the name implies, the TransformGroup is for uniformly translating, scaling, and rotating child nodes. In our example, a TransformGroup is used on the view branch to allow the virtual camera to be moved in the scene.

Figure 2 shows additional objects that make up the view branch, which are beyond the scope of this article. Luckily, Java3D has a SimpleUniverse utility class so it's easy to create the universe and view branch. The SimpleUniverse object allows you to add your content branch and access the view transform group to change the view.

The content branch contains the shapes and lights in your scene. User interaction with the scene is accomplished by adding behaviors to the content branch. A behavior links keyboard, mouse, or temporal events with changes to the content or view. For example, the keyboard or mouse can be used to update the view transform, allowing the user to move the virtual camera through the scene. Time can be used to animate the movement or morphing of shapes in your scene. For our purposes we'll use a modified version of the Java3D KeyNavigator behavior to move over the surface of Mars. There are many other predefined behaviors in Java3D and you can create your own as well. The following code shows how to build the universe and scene.



**Figure 1** A Mars landscape rendered with Java3D



```

...
1  canvas = new Canvas3D(config);
2  universe = new SimpleUniverse(canvas);
3  BranchGroup content=createSceneGraph();
4  addBehaviors(content);
5  addLights(content);
6  content.compile();
7  universe.addBranchGraph(content);
// The scene is ready to render

```

This code includes the steps taken in the Java3DApplication class included in the source code. (The source code for this article can be downloaded from [www.sys-con.com/java/sourcecc.cfm](http://www.sys-con.com/java/sourcecc.cfm).) This class implements a common recipe for Java3D applications.

### Shaping Up

Visual objects in Java3D are represented in the content branch by instances or subclasses of Shape3D. A shape maintains references to an appearance and geometry. Java3D considers the appearance and geometry to be node components. Node components are associated with tree nodes through a reference relationship as shown in Figure 2. Nodes in the scene graph determine what is rendered while node components determine how the nodes are rendered. Node components are not technically considered part of the scene graph because multiple shape nodes can share an appearance or geometry.

The geometry object determines the topology of the shape. There are several utility geometries available to create cubes, spheres, cylinders, and cones. These utility geometries are useful for learning, prototyping, or even assembling larger composite visual objects. More complex geometries can be created with subclasses of GeometryArray that organize arrays of triangles, lines, points, or quadrilaterals. Creating geometries with the GeometryArray subclasses is complex, but thankfully the GeometryInfo utility class drastically simplifies the process. We'll use the GeometryInfo class with a QuadArray to create the landscape geometry for Mars.

The appearance object is used to describe the color, material, polygon, texture, and transparency attributes of a shape (see Figure 3). A ColoringAttributes object defines the color of the shape in an unlit scene as well as the type of

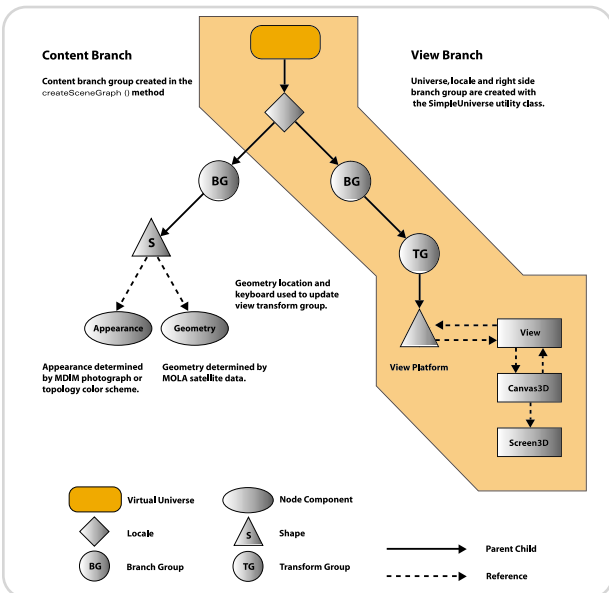


Figure 2 Mars visualization scene graph design

shading to use (i.e., flat or smooth). The Material object specifies the color of the entire shape in a lit scene as well as the shininess of the shape. The PolygonAttributes object is used to render the shape as points, a wire frame, or a solid, and determine which polygons are to be culled. The TextureAttributes object controls how textures are combined with the shape color if the shape is textured. Texturing is valuable when there is a need to present a higher level of detail than what is available with the geometry. As you might have guessed by now, the TransparencyAttributes object helps you render objects such as glass or water.

Now that you have your shape defined with a geometry and appearance, all you have left to do is add the shape to the content branch group. This is done by using the addChild() method on the branch group.

### Shedding Light on Your Scene

Very simple scenes don't require lighting; however, if we are to render a realistic landscape, lighting is very important. Java3D comes through with support for ambient, directional, point, and spotlights. Ambient light has no location or direction but still illuminates the scene. Directional lights have no location but have a direction specified by a vector. Point lights provide attenuated light from a position and illuminate in all directions. Spotlights also provide attenuated light from a location as well as direction with added control of the spread angle and concentration. For all types of lights you can specify the color and the region of influence. A region of influence is the volume of space in the scene that the light affects. Adding lights to a scene is as easy as using the addChild() method on the content branch group.

**Build Incredible Interactive Diagrams with JGo™**

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

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

Learn more at:  
[www.nwoods.com/go](http://www.nwoods.com/go)  
 800-434-9820

Northwoods SOFTWARE CORPORATION

A typical scene uses ambient light combined with two directional lights. Our Mars landscape uses ambient light and two spotlights.

### Creating the Landscape of Mars

With these Java3D concepts under our belt, it's time to turn our attention to creating content. We need a way to create realistic geometry and appearance objects for a Mars landscape. This is where we can use publicly available data to do both. Several successful Mars missions have accumulated terabytes of photos and data. The Mars Global Surveyor (MGS) was one of those missions that carried several scientific instruments.

One of the instruments aboard the MGS was the Mars Orbiter Laser Altimeter (MOLA). If you've been to a home improvement store lately you'll conceptually understand how this instrument works. The latest home improvement gadget is a laser measuring tape. You point the gadget at a wall and it precisely measures the distance between you and the wall. The MOLA took over one billion measurements while the MGS circled the planet for 2.5 years. At the highest precision, those results were used to create a huge matrix of altitude measure-

ments every 0.00781 degrees of longitude and latitude. This high precision data can detect objects as small as 460 meters, making it ideal for major terrain features, but you won't see the rock-level details seen in rover photographs. Today this data is used by NASA for mission planning and is available for the public to download as MOLA Mission Experiment Gridded Data Records (MEGDR). The MEGDR data is available at other resolutions as well, providing us with a wealth of geometry data.

The MOLA data can be used with a Java3D QuadArray object to create an accurate Mars geometry. The MOLA data is organized as a large two-dimensional matrix with the longitude and latitude serving as the dimensions. The value at a given longitude and latitude is the average altitude for that area. A QuadArray object can be used to draw an array of vertices as individual quadrilaterals using a group of four vertices to define a quadrilateral. We'll use four neighboring MOLA data elements as vertices to create a series of quadrilaterals across an area of interest. The geometry of the planet surface is created by mapping the longitude, latitude, and altitude of the MOLA data to three-dimensional points needed for the vertices of the QuadArray. The following code uses GeometryInfo to easily create a geometry object.

```

...
1  GeometryInfo gi =
    new GeometryInfo(
        GeometryInfo.QUAD_ARRAY);
2  ...
    // Map MOLA data to 3D points in
    // Point3d[] coordinates variable
3  gi.setCoordinates(coordinates);
4  // Other texture or coloring tasks
5  NormalGenerator ng = new
    normalGenerator();
6  ng.generateNormals(gi);
7  Geometry g = gi.getGeometryArray();

```

Once the MOLA data is converted into three-dimensional points, a NormalGenerator can be used to finish the geometry object. A normal is a unit vector that defines the orientation of a surface such as a quadrilateral or triangle. Java3D uses the normal of a surface for hidden surface removal and to render the lighting effects of visible surfaces. The NormalGenerator computes the normal vectors for the quadrilaterals in the QuadArray and stores the results in the GeometryInfo object. You may have noticed that we skipped over line 4. This is where some impressive effects can be implemented with Java3D: vertex coloring and texture mapping.

#### Vertex Coloring

The Jet Propulsion Lab (JPL) and the MOLA Science Team have published numerous color-coded topographical maps of Mars. Colors are used to signify altitude ranges with colors blending nicely from one altitude to the next. This technique can be replicated in Java3D by using the GeometryInfo and vertex coloring. Java3D uses the Material object of a shape to determine the color unless the geometry has colors assigned. Colors are assigned to each vertex by first determining the color (based on altitude in this case) for each vertex. The array of colors is then set on the GeometryInfo object by using the setColors() method. The JPL color scheme can be reproduced by using vertex coloring in combination with smooth shading to produce results similar to Figure 4. Other available data such as soil element composition can be used to overlay hematite deposits on your landscape. You could do this with a special vertex color at the location of the hematite deposits or create an image and overlay the image using texture mapping.

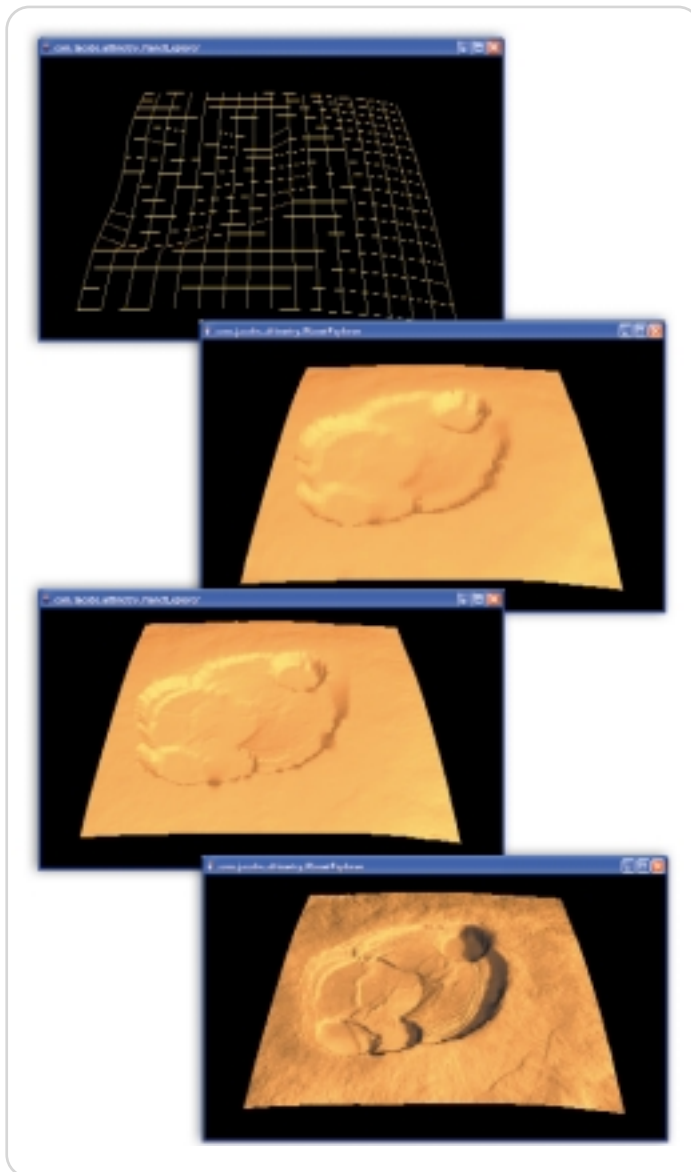


Figure 3 The effect of using different appearance options (wire frame, flat, smooth and textured)



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



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

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

## You need DiagnoSys.

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

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

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



[www.hwcs.com](http://www.hwcs.com) | 1.800.338.6692

### Texture Mapping

A photograph is the most striking and realistic overlay. There are literally thousands of pictures of the surface of Mars but we need undistorted pictures of the entire planet. Luckily, the United States Geographical Survey (USGS) Astrogeology Research Program has gone to the trouble of creating precisely what we need. They have taken Viking 2 era photographs and processed them to eliminate spherical distortions and aligned the resulting photograph mosaics with landmarks in the MOLA data. The result is called the Mars Global Digital Image Mosaic (MDIM), and includes 30 files covering the planet in a form we can use, except for the poles. The highest-resolution photographs are four times more detailed than the MOLA data, but still represent about a football field per pixel. Future releases of MDIM will include the poles and even more higher-resolution photographs.

The number of files in the MDIM poses a challenge to allow the rendering of any location. The MDIM is divided into regions based on the USGS Mars Chart (MC) series of printed maps. The MC regions form a staggered grid and the most useful, high-resolution images are far too large to create one large composite image. The solution is to use the imaging support provided by ImageIO, BufferedImage, and AffineTransform objects to create our own personal mosaic of the area to be rendered. Once this image is created it can be used with Java3D as a photographic overlay.

As Figure 1 shows, Java3D supports the draping of a photograph over the geometry through texture mapping. Java3D automatically renders the photograph as a texture, provided we tell Java3D how to map the pixels in the texture to the vertices in the geometry. Java3D uses a simple texture coordinate system that specifies the horizontal and vertical texture coordinates as values ranging between zero and one. Similar to how we demonstrated vertex colors, texture coordinates must be determined for each geometry vertex. An array of texture coordinates is then set on the GeometryInfo object by using the setTextureCoordinates() method. To finish the appearance of the landscape, the texture and how to combine it with the geometry color is set in the TextureAttributes object. There are many ways to combine the texture color with the geometry color, but we'll use the MODULATE option. This option takes the product of the texture and geometry colors to create the final color. Colors in Java3D use the RGB model with each color component ranging between zero and one. Because the Mars photographs are black and white, the resulting product is a very realistic coloring of the landscape.

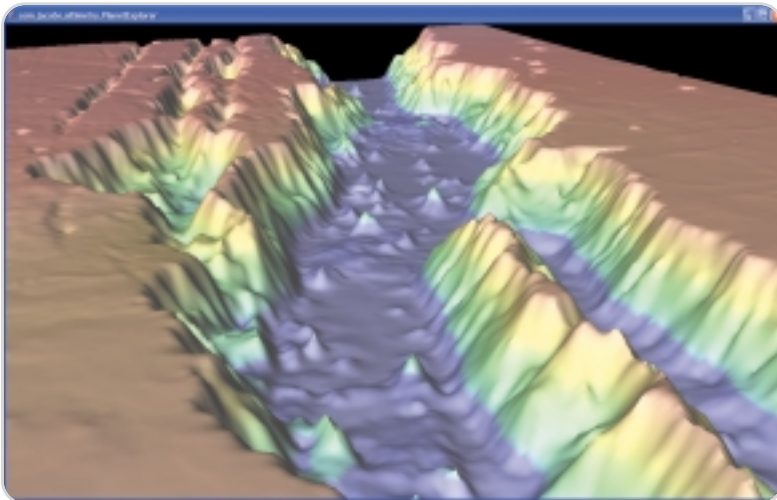


Figure 4 Vertex coloring and smooth shading at work

### Finishing Touches

The appearance of the landscape looks great, but environmental effects such as clouds and a sun will make it even better. Java3D supports enough tools to fake a great atmosphere including a fog shape that can be any color. Using a gray fog will give our Mars landscape more apparent depth. A black fog can be used to create an eerie fade to a dark unknown. The fog does not interact with lights in Java3D nor does the fog have volume, both of which would allow us to create clouds. An atmosphere with clouds takes a bit more creativity, but we have already explored everything you need to know to create a reasonable atmosphere.

The atmosphere in Figure 1 is accomplished by surrounding the scene with a huge textured sphere. The sphere is one of the geometries supplied by Java3D that can automatically create normal vectors and texture coordinates. Simply provide the color and texture in an appearance object and Java3D combines them to create a sky sprinkled with wispy clouds. The atmosphere comes alive when we light it up with a sun.

Our Mars landscape uses ambient light and two spotlights to create the sun in the dusty atmosphere. Here is where we can take advantage of a missing feature of Java3D. Because of the computational requirements, Java3D does not automatically render true shadows. Consequently, light passes through objects and fully lights the area where shadows would naturally occur. The sun can be implemented with a spotlight just outside of the atmosphere sphere. Java3D renders the concentrated light on the sphere as a circle and light passes through the sphere to light the planet surface below. A second spotlight with a wider spread is used to create an aura of the sun. The lighting effects of the spotlights on the atmosphere sphere give the impression of a sun shining down on the landscape.

### Explore Away

Java3D is a powerful programming interface able to handle many 3D programming tasks. This article scratched the surface of what is possible, so while you explore the Mars surface think about using Java3D for GIS studies, medical imaging, 3D puzzle games, or educational programs. The sky is no longer the limit. ☺

### References

- *Java3D*: <http://java.sun.com/products/java-media/3D/index.jsp>
- *Mars Global Surveyor Mission*: <http://marsprogram.jpl.nasa.gov/mgs/>
- *Mars Orbiter Laser Altimeter*: <http://marsprogram.jpl.nasa.gov/mgs/sci/mola/mola.html>
- *MOLA Science Investigation*: <http://ftpwww.gsfc.nasa.gov/tharsis/mola.html>
- *MOLA MEGDR archive*: <http://pds-geosciences.wustl.edu/missions/mgs/megdr.html>
- *Mars Digital Image Mosaic*: <http://astrogeology.usgs.gov/Projects/MDIM21/>
- *Mars Locations*: <http://planetarynames.wr.usgs.gov/mare/mareTOC.html>
- *Java3D Resources*: [www.j3d.org](http://www.j3d.org)

### Acknowledgments

- MDIM photographs are in the public domain and are courtesy of the USGS Astrogeology Research Program.
- MOLA data courtesy of the PDS Geosciences Node of Washington University in Saint Louis.
- Thanks to the MOLA Science team (Neuman) for their help in understanding the JPL topographical color scheme.

# BPP: The Beanshell Preprocessor

by Warren MacEvoy

Increase your productivity benefits

The Beanshell preprocessor, or BPP for short, is intended to be a convenient and powerful preprocessing tool for Java developers. It's convenient because the preprocessor is based on Beanshell, which is essentially interpreted Java. This means that Java or Beanshell programmers can quickly use all of BPP's features. It's powerful for the same reasons: all the power of the Java SDK with the convenience of the Beanshell scripting language is available as double payment: once as a development language and once as a preprocessing language.

In the current world of sexy words for new concepts in software development, a new preprocessor is sure to draw a yawn. But for Java developers, this preprocessor can revolutionize core software development. The reason is that it isn't actually just a new processor, but an entirely different kind of preprocessor: a symmetric preprocessor. A language with a symmetric preprocessor gives the full power (and syntax) of the language as a preprocessor, including the provision for a preprocessor, and so on.

## Getting BPP and Beanshell

BPP is available as an executable JAR file from <http://bpp.sourceforge.net>. Beanshell is available from <http://beanshell.org>. BPP uses Beanshell under the hood and so must find its JAR file in the class path. Putting the bsh-2.0b1.jar in your classpath ensures it will be found. Once this has been set up correctly, use the command:

```
java bsh.Console
```

from a command shell to start up the Beanshell desktop. A windowed Beanshell desktop should appear. In the window, type:

```
you="Jay R. EE";
print("Welcome, " + you);
```

After seeing the expected message, add a sticky note to your monitor with the words: "Learn Beanshell now; save time

later." For now, just close the Beanshell desktop (see Figure 1). BPP does not use the Beanshell desktop; instead it uses the bsh.Interpreter class internally as a lightweight Java interpreter.

## Show Me How Useful BPP Is!

BPP allows you to write Java code that writes Java code in a convenient way. The preprocessor lines begin with a # sign and are executed at preprocess time, while normal lines are undecorated. \$id and \$(expr) on otherwise normal lines are translated in a natural way. For example, the following BPP source file will create the traditional "Hello World" Java program, but with the message appearing in four different languages.

```
#
# greetings=new String[] {
#   "hola mundo",
#   "ciao mondo",
#   "hello world",
#   "\u043F\u0440\u0438\u0432\u0435\u0442
\u043C\u0430\u0438\u0440",
# };
#
public class Xample1 {
    public static void main(String[] args) {
        #for (i=0;i<greetings.length; ++i) {
            System.out.println("$(greetings[i])");
        }
    }
}
```

If you save the above code in a file named Xample1.java.bpp, then running BPP with the line

```
java -jar bp.jar Xample1.java.bpp
```

will produce the following text in Xample1.java:

```
public class Xample1 {
    public static void main(String[] args) {
        System.out.println("hola mundo");
        System.out.println("ciao mondo");
        System.out.println("hello world");
        System.out.println("spasey wap");
    }
}
```

The last line is "hello world" in Russian, and may appear strangely on

systems that don't understand UTF-8 encoded unicode. Point is, the #ed lines are executed at "preprocess time" by BPP. This results in a Java source file with, in compiler optimization parlance, an "unwound loop."

Admittedly, unwinding the above loop at preprocess time will provide no particular advantage over executing the loop at runtime. There are places where such unwinding could make a great deal of difference. Similar preprocessor code could write substantial "boiler plate" code that you might otherwise use a more traditional "copy/paste/edit" approach on, but we will leave that to the reader's imagination (and online tutorials on the BPP Web site). The next example takes on a loftier software engineering goal: compile-time versus runtime safety.

## Safe Sets

It's a basic principle of software engineering that the earlier you find an error, the less expensive it is. The collections framework in Java is a good example of something that's runtime safe (because you can't put in an apple and treat it like an orange without a ClassCastException), but it isn't compile-time safe, since you can add any object to any collection, even if you only really wanted oranges in it.

BPP can quickly create compile-time type-safe wrapper classes for collections (or wherever else you need them). Listing 1 provides a snippet of the type-safe template for Collection.

To use the code in Listing 1, save it

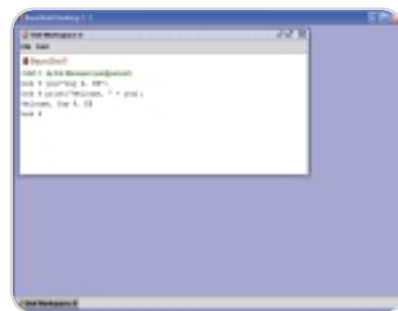


Figure 1 Beanshell desktop

**Dr. Warren MacEvoy** has been an editor of corporate Java training manuals and lead Java instructor for Sun Java programmer certification programs. He has been a programmer and educator since time immemorial (or at least since 1995). He is the Rocky Mountain Regional director of the ACM intercollegiate programming contest, which is his main contribution to fun for young software engineers.

wmacevoy@mesastate.edu

in a file called “makeTypedCollection.bpp” and generate the equivalent Beanshell script with:

```
java -jar bpp.jar -b makeTypedCollection.bpp
```

The `-b` option tells BPP to create the script, but not execute it. The script is written in this case to `makeTypedCollection.bsh`. With this handy “template” around, creating a compile-time type-safe collection is a snap. For example, a string collection would be the following smidgen of lines in `StringCollection.java.bp`:

```
#source("makeTypedCollection.bsh");
#makeTypedCollection("StringCollection","java.lang.String");
```

Notice that I source the Beanshell script generated by BPP, not the BPP script itself.

There is a philosophical point here: judicious use of BPP can move many checks from runtime to compile time. This can make an order-of-magnitude difference in how long it takes (and expensive it is) to find and correct mistakes. Now let's look at how BPP does its magic.

## How Does It Work?

BPP is similar in nature to servlets and php (and a very similar Perl-based pre-

processor, `perlpp`). It works by translating the BPP code into Beanshell code under the following rules:

- Lines with a # (pound) in column one have the remainder of the line copied exactly to the Beanshell script. For example:

```
#n=10;
#for(i=0; i<n; ++i) {
```

becomes

```
n=10;
for(i=0; i<n; ++i) {
```

in the Beanshell script.

- Lines with a " (double quote) in column one have the remainder of the line “quoted magically.” This means it becomes a print statement with `$IDENTIFIER` and `$(EXPRESSION)` patterns concatenated in. Two compromises were made in the magic translation:
  - `$` is a legal start and part of a Java identifier. If you need to have such a value in a BP script use `$(my$-strange$Id)`.
  - When `$` is not succeeded by an identifier, a left parenthesis, or another `$`, it simply represents a single `$`. `$$` on a magically quoted line represents a single `$`. For example:

```
#static import bpp.Format.*;
"Dear $title $lastName;
"You owe $$$$(N(blnc,"#",###.00")).
```

becomes

```
static import bpp.Format.*;
print("Dear "+title+" "+lastName+");
print("You owe $$$$(N(blnc,"#",###.00))+".");
```

in the Beanshell script. For those new to Beanshell, `print()` is equivalent to `System.out.println()`. The `N(Number n,String f)` method is a static member of the `bpp.Format` class as a convenience for formatting numbers.

- Lines with a ' (single quote) in column one have the remainder of the line ‘quoted exactly.’ This generates a print statement in the Beanshell script that will faithfully reproduce the line of text. For example:

```
'#static import bpp.Format.*;
'Dear $title $lastName;
'You owe $(N(blnc,"$#",###.00)).
```

becomes

```
print("#static import bpp.Format.*;");
print("Dear $title $lastName;");
print("You owe $(N(blnc,\"$#",###.00\")).");
```

in the Beanshell script.

- Lines with none of the above “translation codes” have the default translation applied to them. This is “quoted magically” unless the `-q` (‘quote exactly by default’) option is passed to BPP.

Putting these together for the first example, the first BPP source file, `Xample1.java.bpp`, produces the Beanshell script:

```
greetings=new String[] {
    "hola mundo",
    "ciao mondo",
    "hello world",
    "\u043F\u0440\u0438\u0432\u0435\u0442
\u043C\u0438\u0438\u0440",
};

print("");
print("public class Xample1 {");
print("    public static void main(String[]
args) {");
for (i=0;i<greetings.length; ++i) {
    print("        System.out.println(\""+greet-
ings[i]+\");");
}
print("    }");
print("");
```

Executing this script with Beanshell generates the promised pure Java source file shown above as `Xample1.java`.

## Conclusion

BPP facilitates versioning, templates, macros, optimizations, and compile-time type safety. These things are a normal expectation of preprocessors. However, because the preprocessor is essentially the same full-featured language as the target language, including the fact that it has a preprocessor, these features are much more accessible than, say, C++ templates are to C programmers. This provides big productivity benefits.

Here are a few other gains.

### Have Fun!

Writing code over and over that's just a little different from the last time is boring. After the third time, you usually see the part that's staying the same and what is changing. With BPP you can codify that and stick to the fun (new) stuff.

### Big Tools Make Little Ones

You can use BPP in development

## You Is What?

Beanshell is a get-to-the-point Java. Typing:

```
you="Jay R. EE";
print("Welcome, " + you);
```

in Beanshell is equivalent to compiling and executing the following Java code:

```
public class SomeClass
{
    public static void main(String[] args)
    {
        String you="Jay R. EE";
        System.out.println("Welcome, " + you);
    }
}
```

Which one would you rather type? See [www.beanshell.org](http://www.beanshell.org) for many useful resources on Beanshell. For now, here is what I claim to be the shortest tutorial of a production programming language in history:

Beanshell is an interpreted version of Java with optional types.

Types are great for safety, but they can turn the quick and dirty into the slow and tedious. Beanshell adheres to types you specify, but also allows unspecified types so you can choose your safety level. As of version 2.0, Beanshell has seamless integration with JDK 1.3 and above. Thus Beanshell is a flexible superset of Java that I encourage any Java developer to get to know.

Dr. Piet Jonas in his "Type Safe Collections" article also addresses type safety, but in another manner.

Jonas' idea is to verify the types for collections at insertion time using runtime type information. This causes an exception to be thrown early (at insertion time) rather than late (at extraction time). In either case, the exception is thrown at runtime.

The type-safe wrappers we suggest using here allow for compile-time safety.

and take advantage of the latest and greatest JDKs in your development environment to produce solutions in any target language/architecture. BPP is used in this way to support a multilanguage environment called FRAMES.

### Protecting IP

The JavaBean model reduces an uber-model to one that solves a particular problem. Giving such a bean to someone else allows them to reverse-engineer your IP. BPP can generate a specific solution to a problem without revealing any general techniques on how that specific solution was constructed. Karl Castleton is writing a BPP-based tool to generate 80% of the boiler plate code for a Java servlet/ MySQL Web site based on an XML description of the database architecture. This includes the SQL initialization, basic form pages, and compile-time-safe SQL access classes.

### Simplification

In both the above cases, the code that BPP writes is as readable as what a specific programmer would have written to solve the same problem. For example, the above SQL framework generates code that a Java programmer who knows nothing about BP XML, or even SQL can easily use. In the FRAMES application, BPP writes specific documentation appropriate for each supported target language based on a single document root.

Enjoy BPP! ☺

### References

- Beanshell: [www.beanshell.org](http://www.beanshell.org)
- BPP: <http://bpp.sourceforge.net>
- FRAMES (click on the "FRAMES" link): <http://mepas.pnl.gov/earth>
- BPPSql: <http://home.mesastate.edu/~kcastle/BPPSql.html>
- perlpp: <http://tdp.org/LDP/LG/issue44/macevoy/macevoy.htm>

#### Listing 1

```
#void makeTypedCollection(String className,String element) {
public class $className {
    public static class $(className)Iterator {
        java.util.Iterator iterator;
        public Iterator(java.util.Iterator _iterator) {
            iterator=_iterator; }
        public boolean hasNext() { return iterator.hasNext(); }
        public $element next() { return ($element)iterator.next(); }
        public void remove() { iterator.remove(); }
    }
    protected java.util.Collection collection;
    public $className(Collection _collection) { collection=_collection; }
    public boolean add($element arg1) { return collection.add(arg1); }
    public boolean contains($element arg1) { return collection.contains(arg1); }
    public boolean remove($element arg1) { return collection.remove(arg1); }
    public $(className)Iterator iterator() {
        return new $(className)Iterator(collection.iterator());
    }
}
#} // makeTypedCollection
```

Advertiser	URL	Phone	Page
Agitar	<a href="http://www.agitar.com">www.agitar.com</a>	650-694-7572	37
Altova	<a href="http://www.altova.com">www.altova.com</a>	978-816-1600	7
AMD	<a href="http://developer.amd.com">developer.amd.com</a>	408-749-4000	45
Borland	<a href="http://www.go.borland.com/j6">www.go.borland.com/j6</a>	831-431-1000	9
Business Objects	<a href="http://www.businessobjects.com/v10/047">www.businessobjects.com/v10/047</a>	800-877-2340	27
Canoo Engineering AG	<a href="http://www.canoo.com/ulc">www.canoo.com/ulc</a>	41 (61) 228 94 44	15
ClearNova	<a href="http://www.clearnova.com/thinkcap">www.clearnova.com/thinkcap</a>	770-442-8324	31
DataDirect	<a href="http://www.datadirect.com">www.datadirect.com</a>	800-876-3101	Cover II
Dice	<a href="http://www.dice.com">www.dice.com</a>	877-386-3323	39
Dralasoft Corp.	<a href="http://www.dralasoft.com/customers">www.dralasoft.com/customers</a>	303-468-6754	3
Enerjy	<a href="http://www.energy.com">www.energy.com</a>	866-598-9876	4
Fiorano	<a href="http://www.fiorano.com/downloads">www.fiorano.com/downloads</a>	800-663-362	47
Google	<a href="http://www.google.com/cacm">www.google.com/cacm</a>	650-623-4000	35
GreenPoint	<a href="http://www.webcharts3d.com/demo">www.webcharts3d.com/demo</a>	212-765-6982	41
H&W Computer Systems	<a href="http://www.hwcs.com/wldj2.asp">www.hwcs.com/wldj2.asp</a>	800-338-6692	59
ILOG	<a href="http://jviews-info-kit.ilog.com">jviews-info-kit.ilog.com</a>	1-800-for-ILOG	17
Inferdata	<a href="http://www.inferdata.com">www.inferdata.com</a>	888-211-3421	49
InstallShield	<a href="http://www.installshield.com/define">www.installshield.com/define</a>	847-466-4000	25
JavaOne	<a href="http://www.java.sun.com/javaone/sf">www.java.sun.com/javaone/sf</a>	888-886-8769	65
Librados	<a href="http://www.librados.com">www.librados.com</a>	888-341-4258	11
Northwoods Software Corp.	<a href="http://www.nwoods.com/go">www.nwoods.com/go</a>	800-434-9820	57
Oak Grove Systems	<a href="http://www.oakgrovesystems.com">www.oakgrovesystems.com</a>	818-440-1234	43
Parasoft Corporation	<a href="http://www.parasoft.com/jtest">www.parasoft.com/jtest</a>	888-305-0041	53
Parasoft Corporation	<a href="http://www.parasoft.com/soapstest">www.parasoft.com/soapstest</a>	888-305-0041	23
Quest Software, Inc.	<a href="http://www.quest.com/jdj">http://www.quest.com/jdj</a>	800-663-4723	Cover IV
Rascal Software	<a href="http://www.rascalsoftware.com/jdj">www.rascalsoftware.com/jdj</a>	206-624-7300	21
ReportingEngines	<a href="http://www.reportingengines.com">www.reportingengines.com</a>	888-884-8665	33
SESMA	<a href="http://www.sesma.com/jdj">www.sesma.com/jdj</a>	949-559-SESMA	55
Sleepycat Software	<a href="http://www.sleepycat.com/bdbje">www.sleepycat.com/bdbje</a>	510-597-2128	29
Software FX	<a href="http://www.chartfx.com">www.chartfx.com</a>	800-392-4278	Cover III
Visual Paradigm	<a href="http://www.visual-paradigm.com">www.visual-paradigm.com</a>	852-820-1912	19
WebAppCabaret	<a href="http://www.webappcabaret.com/jdj.jsp">www.webappcabaret.com/jdj.jsp</a>	831-464-6941	51
Web Services Edge 2005 East	<a href="http://www.sys-con.com/edge">www.sys-con.com/edge</a>	201-802-3069	66

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

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



Onno Kluyt

# From Within the Java Community Process Program

## The technology evolution

**W**elcome to the June edition of the JCP column! Each month you can read about the Java Community Process: newly submitted JSRs, new draft specs, Java APIs that were finalized, and other news from the JCP. This time we'll discuss a long-running JSR that has successfully finished, a new version of the J2EE platform that has been proposed, as well as JavaServer Pages, but first we'll start with some number crunching.

### Are We Getting Better At It?

The JCP has been in operation a little over five years. During this period the community has started just over 240 JSRs

group, and how to gain and capture developer feedback. With JCP 2.6 it's now even easier to receive and give feedback; let's see how fast the technology evolution can really go!

### The J2EE Technology, Version 1.5

New on the roll call is JSR 244, which proposes the next version of the J2EE platform. Ease of development is the main theme for this version. As the umbrella JSR for the server-side platform, it brings together several JSRs already well underway, such as JSR 181 (Web Services Metadata), JSR 220 (EJB 3.0), JSR 222 (JAX-B 2.0), JSR 224 (JAX-RPC 2.0), JSR 127 (JavaServer Faces),

These and other works will be undertaken in close consultation with the JavaServer Faces expert group.

### Public Review

The Content Repository JSR led by Day Software has entered Public Review. This effort has migrated to the new JCP 2.6 process; this means the SE/EE Executive Committee will be voting on this JSR during the last week of the review. This specification defines content services that can be performed on a content repository. Examples are author-based versioning, full textual searching, access control, and event monitoring.

“The speed with which JSRs now complete is significantly higher: JSRs now enter Public Review an average of 100 days sooner than before and finish on average 200 days earlier”

and finished a third of those. Over these five years the JCP has changed considerably with the installment of the Executive Committees, individual membership in addition to company membership, and the ability to use open source software licenses. This raises the question: What impact do these changes have on the speed with which the community process delivers new specifications? In the first half of its life the JCP finalized about the same number of JSRs as during the last two years (40 versus 44). The speed with which JSRs now complete is significantly higher: JSRs now enter Public Review an average of 100 days sooner than before and finish on average 200 days earlier. I believe this is as much a result of process improvements as it is of a more experienced community: many of today's spec leads have led JSRs before so they know how to work with an expert

and JSR 52 (JSTL 1.1). The JSR is proposing an aggressive schedule that would see this version completed in the second half of 2005.

### JSR 245 - JavaServer Pages 2.1

As the submitter for the J2EE 1.5 platform JSR notes, some modifications are needed in this API so that there will be a smooth integration with JavaServer Faces. This JSR proposes to further enhance the JSP 2.0 Expression Language. Some of the enhancements it will be considering are:

- Putting the expression language chapter into its own specification document
- The ability to plug in variable resolvers and property resolvers
- The ability to express references to bean methods and bean properties
- The ability to defer expression evaluation

### JSR 75

PalmSource and IBM, co-spec leads, successfully navigated the PDA Optional Packages for J2ME JSR in and through the Final Approval Ballot. While time-wise it was a bit of a journey for this JSR, it's now been completed and is available for implementation and distribution. The JSR is meant to be used on top of CLDC environments. It consists of two optional packages: Personal Information Management (PIM) for management of items like calendar, address book, and to-do lists, and the FileConnection package that provides access to a device's file system including external memory cards and the like.

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

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

onno@jcp.org



June 28–  
July 1, 2004  
Moscone Center  
San Francisco, CA



Image credit: NASA/JPL/California Institute of Technology

Benefit from four days of in-depth training and networking with the foremost leaders and creators of Java™ technology in:

- Enterprise
- Mobility
- Web Services
- Desktop



## Go Anywhere You Want with Java™ Technology

It all starts at the JavaOne<sup>SM</sup> conference.

Java™ technology is everywhere, improving the digital experience for everyone. It all starts at the JavaOne<sup>SM</sup> conference, your source for cutting-edge knowledge and proven solutions. You'll learn how to tap into the full power of the Java platform: platform independence; compatibility with and between editions; open toolsets and technologies supported by a global developer community.

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

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

# JavaOne<sup>SM</sup>

Sun's 2004 Worldwide Java Developer Conference™

**Save \$100!** Register by June 27, 2004, and receive \$100 off the full Conference package. Priority code: ADARZKND

Register at  
[java.sun.com/javaone/sf](http://java.sun.com/javaone/sf)

Sponsored by



Produced by



Copyright © 2004 Sun Microsystems, Inc. All rights reserved. 0000164. Sun, Sun Microsystems, the Sun logo, Java, the Java Coffee Cup logo, JavaOne, the JavaOne logo, Java Developer Conference, all Java-based marks and logos, and J2SE are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.

# Job Discernment



Jason Bell

Those of you kind enough to read my editorials for *JDJ* would have noticed that I started a new job. A fresh start, a new year, a colossal waste of my time it turned out. Startup companies can be odd to work for sometimes and you have to read between the lines when it comes to statements from directors and shareholders. To cut a short story shorter than the original, I was laid off after four weeks and given the excuse that my math was not at PhD level, which is interesting as it was never mentioned during the five hours of interviews and pair programming exercises. In the UK if you have worked less than 12 weeks with an employer, they only have to serve you with one week's notice.

With a new wave of startups coming again with the possible tech recovery, I present to you my example of what not to do.

Regardless of who you are, what you do, or what you get paid you must get everything in writing. And I mean everything. Get your employment contract sorted out before you start, not within the first couple of weeks on the new job. Also, take time to read your contract carefully and don't be afraid to ask questions. In terms of a probationary period, at a new employer I would seriously negotiate a fixed time before the employer can serve notice.

I am always eager to get feedback from my new peers so I ask a lot of

questions about how they think things are going. The responses I received were always "You're doing brilliant" or "Excellent" with huge amounts of enthusiastic looks. The next week I was served one week's notice and a very cold shoulder.

Most employment recruitment is done on the basis of your history. You arrive at your interview with your résumé and wow them at the interview and then you get started. You spend an awful amount of time convincing the company why they need you. Now it's time to get the company to explain why you need them. Sounds simple, doesn't it? It's overlooked the majority of the time. Employers have one goal in mind: "to maximize long-term shareholder value by the selling of products and services"\*; the company is not there solely for your career development. Training programs and personnel mentoring are all very well but the company has one main aim in mind – to make money. If you find a company is being overly keen to help you beyond all reason, start asking yourself some serious questions. Most of the training I have done has been on my own time in addition to what I've learned from completing the task in hand.

Take time to research the company. Look at their Web site and if they don't have one, ask why. Check for the existence of proper e-mail services to be sure the company is not running all

their communication from a free Web mail account. If you have questions, ask them. Listen to the answers very carefully; this can be an indication of how things are really going for the company.

Your outside interests should not dictate how you are going to perform within a company. It became apparent that my complete lack of interest in role-playing games was proving to be a problem as that was the only way that the rest of the team would socially interact.

Finally, keep an open mind. Though all things may look completely left of center, the company you are working for may just be a diamond waiting to shine. Looking back I'm happy that the event happened as it spurred me on to get a grip and get some other projects completed. It also made me seriously look at the network of contacts that I have and how to maximize them. Not to use them so much that they hate the sight of your e-mails, but to talk to them, and see what the industry is doing. Share what you are doing and how things are going. Developers are suppliers to an extremely competitive industry and how we act professionally will ultimately determine which way our industry will head. ☺

## Reference

- \*Sternberg, E. (2000). *Just Ethics, Business Ethics in Action*. Oxford Press.

Jason Bell is the founder of [www.bigpharmanews.com](http://www.bigpharmanews.com) and is also involved with the development of RSSLib, an API for writing and reading RSS files.

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

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

web services **EDGE**  
conference & expo

Web Services Edge 2005 East  
International Web Services Conference & Expo

## Call for Papers Now Open!

[www.sys-con.com/edge](http://www.sys-con.com/edge)

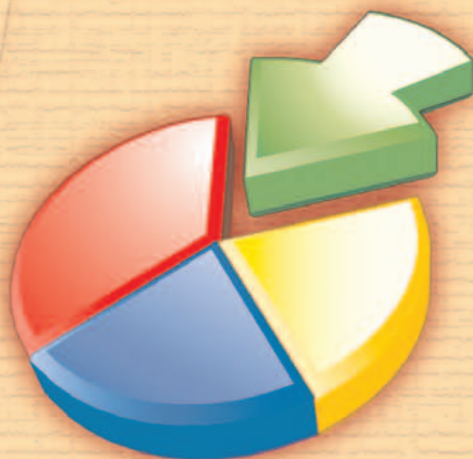
The Largest *i*-Technology Event of the Year!  
Guaranteed Minimum Attendance 3,000 Delegates...

Tuesday, 2/15: Conference & Expo  
Wednesday, 2/16: Conference & Expo  
Thursday, 2/17: Conference & Expo

PRODUCED BY  
SYS-CON  
EVENTS

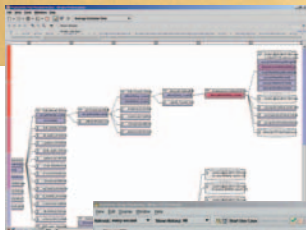
# FREE JAVA CHARTS!

Download the Chart FX for Java Community Edition now.



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

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



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



JProbe – a performance tuning toolkit for Java developers.

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

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

