

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

JDJ

WWW.SYS-CON.COM/JDJ VOL.10 ISSUE:6

**WHAT'S NEW IN
ECLIPSE 3.1**

PAGE 30

STAR TREK

TECHNOLOGY FOR JAVA3D

Building a particle system for Java3D

RETAILERS PLEASE DISPLAY
UNTIL JULY 31, 2005

\$5.99US \$6.99CAN

07>



PLUS...

- ▶ **Java Application Security
in the Corporate Word**
- ▶ **Coexisting in the
Java Universe**
- ▶ **Kicking the Tires
on Java 5.0**

Let your ideas run free.

With Visual Studio®.NET 2003, you write less code and achieve higher productivity, so you can turn that big idea into reality faster than you ever thought possible. You get a powerful and efficient IDE, support for Windows® and Web Forms that eliminate hundreds of lines of code, and better deployment that makes "DLL hell" an endangered species. Find out more about the tool that helps you take on your biggest ideas: visit msdn.microsoft.com/visual



From JavaOne to JavaTen



Jeremy Geelan



Editorial Board

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

Production

Production Consultant: **Jim Morgan**
 Associate Art Director: **Tami Lima**
 Executive Editor: **Nancy Valentine**
 Associate Editor: **Seta Papazian**
 Research Editor: **Bahadir Karuv, PhD**

Writers in This Issue

Calvin Austin, David Bismut, Ed Burnette, Peter Braswell, Yakov Fain, Marc Fleury, Jeremy Geelan, Mike Jacobs, Philippe Lalande, Murali Kaundinya, Adam Kolawa, Kenneth D. Kruszka, Swaminathan Natarajan, Krishnakumar Pooloth, Ajit Sagar, Venkat, Joe Winchester

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

Subscriptions

For subscriptions and requests for bulk orders, please send your letters to Subscription Department:

888 303-5282
 201 802-3012
subscribe@sys-con.com

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

Editorial Offices

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

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

©Copyright

Copyright © 2005 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 Dorothy Gil, dorothy@sys-con.com. SYS-CON Media and SYS-CON Publications, Inc., reserve the right to revise, republish and authorize its readers to use the articles submitted for publication.

Worldwide Newsstand Distribution
 Curtis Circulation Company, New Milford, NJ
 For List Rental Information:

Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com
 Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

Newsstand Distribution Consultant
 Brian J. Gregory/Gregory Associates/A.W.R.D.S.
 732 607-9941, 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.

Technology birthdays come and go, but Internet technologies, by their very nature, aren't old enough to allow yet for centenaries, or even diamond anniversaries. So it is fascinating to see how people are reacting to the fact that popular technologies like Java, ColdFusion, and Flash have now finally reached – or are about to reach - the ripe old age of 10.

Java was “born” on May 23, 1995. But people forget that RealAudio, too, which allowed us to hear across the Net in real time, dates back to 1995. It was also the year that traditional online dial-up systems like CompuServe, America Online, and Prodigy first began to provide Internet access, and the year that Netscape went public with what was at the time the third largest ever IPO share value on the NASDAQ.

In 1995, The Vatican came online for the first time (<http://www.vatican.va/>) as did the government of Canada (<http://canada.gc.ca/>). The first official Internet wiretap was successful in helping the Secret Service and Drug Enforcement Agency (DEA) apprehend three individuals who were illegally manufacturing and selling cellphone cloning equipment and electronic devices, and Chris Lamprecht (a.k.a. “Minor Threat”) became the first person ever banned from accessing the Internet – by a U.S. District Court judge in Texas.

So when May 23 came this year it was a time for reflecting not just on Java's birth; it was a time for remembering other aspects of its first 10 years too, such as the closing keynote at JavaOne in 1999 by Douglas Adams of *Hitchhiker's Guide to the Galaxy* fame – who as *JDJ's* Calvin Austin remembers in this issue “gave a great perspective on things and not just Java.”

As Ajit Sagar says in his editorial this month, “A lot has happened since the language that was trademarked with

dancing Dukes made its appearance into the world of computing. In its current incarnation, the Java platform is undoubtedly the backbone of distributed enterprise applications in today's IT.” But future possibilities abound. For example, as JBoss's Marc Fleury – whom we are delighted to say has also written for this month's issue – asks: “Why don't we take the underlying concept of EJB 3.0 and apply it to simplify other Java middleware products?”

It is the energies of Java pioneers like Fleury, outside of Sun, that will characterize the next 10 years of Java, just as those of James Gosling, Tim Lindholm and company, within Sun, have helped shape the first 10 years.

May 23 was marked in relatively low-key fashion by Sun, though there

was a symbolic birthday cake

with 10 candles, which stood in front of the inevitable life-size Duke in a ceremony presided over on the Sun campus by Gosling and Sun's president and COO, Jonathan Schwartz.

Duke was originally designed by Sun's Joe Palrang, who was doing

artwork for the UI, and Gosling has explained Duke's appearance – the big hands, the pointed head – as not only representing the personality of Java but also having precise functionality: “We wanted something that could show up on the screen in a relatively small area and yet still be recognizable and comprehensible, something you could put a lot of emotion and gestural activity into, and still be about the size of a postage stamp.”

There will be a much more public celebration at JavaOne; those of you picking up this issue of *JDJ* at the show will experience this for yourselves. Meantime, keep an eye out and see what kind of hijinks will be devised to celebrate the upcoming 10th birthdays, respectively, of ColdFusion (“born” July 1995, just two months after Java) and Flash (1996). ☛

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



Bring your development plans to light

Sneak a peek at XMLSpy® 2005,
the industry leading XML development
environment from Altova®.

Revealed in Version 2005:

- XSLT 2.0 support and debugger
- XPath 2.0 support and analyzer
- XQuery 1.0 support and debugger
- Eclipse integration

See for yourself why XMLSpy 2005 is the standard for modeling, editing, debugging and transforming all XML related technologies. Illuminate your strategy with advanced standards compliance, innovative development and analysis tools, and extended platform integration. Use XMLSpy 2005 to structure XML Schemas and devise XML documents, then automatically generate runtime code in multiple programming languages. Become a markup mastermind!

**Download XMLSpy® 2005
today: www.altova.com**

Join us at
JavaOne,
San Francisco

JDJ contents

JDJ Cover Story

Star Trek Technology for Java3D

Building a particle system for Java3D

by Mike Jacobs

54

FROM THE GROUP PUBLISHER

From JavaOne to JavaTen
by Jeremy Geelan 3

VIEWPOINT

Enterprise Java Made Easy
by Marc Fleury 6

JAVA ENTERPRISE VIEWPOINT

**On the Tenth Year of Java
My Client Asked of Me...**
by Ajit Sagar 8

TECHNIQUES

Shouldn't J2EE Be More Like Java?
Building a better bean
by Kenneth D. Kruszka 10

JVM

**Isolating Concurrent Java Apps
in a Virtual Machine**
Peaceful coexistence
by Murali Kaundinya 22

Q&A

Coexisting in the Java Universe
An interview with Anthony Scotney
Interviewed by Jeremy Geelan 26

CORE AND INTERNALS VIEWPOINT

JavaOne from the Inside Out
by Calvin Austin 38

DESKTOP JAVA VIEWPOINT

The Return of the Client
by Joe Winchester 46

FRAMEWORK

Properties Editor Framework
*Solving the problem of managing
application properties*
by Swaminathan Natarajan, David Bismut, and
Krishnakumar Pooloth 48

LABS

MKS Integrity Suite 2005
Reviewed by Michael Sayko 64

LABS

Jtest
by Parasoft
Reviewed by Venkat 68

JSR WATCH

OSS: The Market Landscape
*Why the OSS industry can – for the first time ever
– produce open, implementable,
and certifiable standards*
by Philippe Lalande 72

Features

18



**Java Application Security
in the Corporate World**
by Adam Kolawa and Yakov Fain

30



What's New in Eclipse 3.1
by Ed Burnette

40



Kicking the Tires on Java 5.0
by Peter Braswell


Marc Fleury

Enterprise Java Made Easy

Simplicity is the key driving force behind the success of Java. When Dr. Gosling invented the Java language in 1995, the goal was to make life easier for software developers. Java's elegant language design, simple API, and vendor-independence have made it the platform of choice for many developers. However, as Java evolves to address enterprise needs for scalability and flexibility, developer friendliness has taken a back seat. The complex programming model in EJB 2.1 and J2EE 1.4 has hindered Java's adoption, and it's the root cause for many slow-performing and error-prone Java applications.

Fortunately, help is on the way. The upcoming EJB 3.0 and J2EE 1.5 servers greatly simplify enterprise Java development without compromising scalability and flexibility. Unlike many other third-party commercial and open source J2EE alternatives, EJB 3.0 is completely standard-based. There is no vendor lock-in. In fact, I think EJB 3.0 is probably the most significant invention in J2EE's history. EJB 3.0 simplifies application development in the following three key areas:

1. *EJB 3.0 eliminates the need for excessive and redundant XML-based deployment descriptors.* Instead, the bulk of configuration options are specified within the source code as Java annotations. The XML deployment descriptors are still available as an optional choice for administrators who need to override default configuration values at deployment time.
2. *EJB 3.0 simplifies Object/Relational Mapping (ORM) via a new entity bean model.* Java developers only need to work with plain old Java objects (POJOs) and build the domain data model, following sound object-oriented design principals. The mapping between object hierarchy and relational table schemas is transparently handled by the EJB 3.0 server. The EJB 3.0 server also manages database connections, enforces transaction rules, generates database-specific

SQL statements, and detects updates to mapped POJOs in the application.

3. *EJB 3.0 enables new application architectures based on the Dependency Injection design pattern.* Resources and services can be declaratively wired into the application via annotations or XML configuration files. That allows developers to build loosely coupled applications.

The JBoss Application Server 4.0.3 is the first J2EE application server to support EJB 3.0. While JBoss EJB 3.0 is still in the beta stage, we know some of our customers are already using it successfully in their production environments. But why stop at EJB 3.0? Why don't we take the underlying concept of EJB 3.0 and apply it to simplify other Java middle-



ware products? That is exactly the direction we are moving toward at JBoss. In the near future, we aim to support the EJB 3.0-style programming model (i.e., annotations, POJO services, and dependency injection) throughout our JEMS (Java Enterprise Middleware System) product suite, includ-

ing JMX, JMS, JSF, jBMP, JBoss Cache, and JBoss Portal. Other Java middleware vendors will like to follow suit and support the EJB 3.0 programming model in their products. Furthermore, the JBoss Eclipse IDE 1.5 integrates EJB 3.0 specific wizards, annotation-aware smart editors, Hibernate code generation tools, and JBoss server management tools, all in one Eclipse-based open source IDE package.

The result of all this is a simpler and more robust J2EE. The ultimate winners, of course, are Java developers like yourself. If you are interested in learning more about EJB 3.0, please come to our JavaOne sessions led by experts such as Bill Burke, Gavin King, Tom Baeyens, Michael Yuan, and Stan Silvert. If you can't make it to JavaOne, you can see much of our EJB 3.0 related content and demos on our Web site at www.jboss.com/javaone05. Hope to see y'all there! ☺

Marc Fleury, PhD, is the founder and CEO of JBoss Inc., the company behind Professional Open Source products such as JBoss Application Server, Hibernate, JBoss Portal, JBoss Cache, and JBoss Eclipse IDE. He started the JBoss.org project in 1999 and co-founded JBoss Inc. in 2002. Dr. Fleury holds a degree in physics from Ecole Polytechnique, a masters in theoretical physics from ENS ULM and a PhD in physics from Ecole Polytechnique for work he did as a visiting scientist at MIT's RLE.

marc.fleury@jboss.org

President and CEO:

Fuat Kircaali fuat@sys-con.com

Vice President, Business Development:

Grisha Davida grisha@sys-con.com

Group Publisher:

Jeremy Geelan jeremy@sys-con.com

Advertising

Senior Vice President, Sales and Marketing:

Carmen Gonzalez carmen@sys-con.com

Vice President, Sales and Marketing:

Miles Silverman miles@sys-con.com

Advertising Sales Director:

Robyn Forma robyn@sys-con.com

National Sales and Marketing Manager:

Dennis Leavey dennis@sys-con.com

Advertising Sales Managers:

Megan Mussa megan@sys-con.com

Associate Sales Managers:

Dorothy Gil dorothy@sys-con.com

Kim Hughes kim@sys-con.com

Editorial

Executive Editor:

Nancy Valentine nancy@sys-con.com

Associate Editor:

Seta Papazian seta@sys-con.com

Production

Production Consultant:

Jim Morgan jim@sys-con.com

Lead Designer:

Tami Lima tami@sys-con.com

Art Director:

Alex Botero alex@sys-con.com

Associate Art Directors:

Abraham Addo abraham@sys-con.com

Louis F. Cuffari louis@sys-con.com

Assistant Art Director:

Andrea Boden andrea@sys-con.com

Video Production :

Frank Moricco frank@sys-con.com

Web Services

Information Systems Consultant:

Robert Diamond robert@sys-con.com

Web Designers:

Stephen Kilmurray stephen@sys-con.com

Percy Yip percy@sys-con.com

Vincent Santaiti vincent@sys-con.com

Accounting

Financial Analyst:

Joan LaRose joan@sys-con.com

Accounts Payable:

Betty White betty@sys-con.com

Accounts Receivable:

Gail Naples gailn@sys-con.com

SYS-CON Events

President, SYS-CON Events:

Grisha Davida grisha@sys-con.com

National Sales Manager:

Jim Hanchrow jimh@sys-con.com

Customer Relations

Circulation Service Coordinators:

Edna Earle Russell edna@sys-con.com

Linda Lipton linda@sys-con.com

JDJ Store Manager:

Brunilda Staropoli brunilda@sys-con.com

ELIMINATE PERFORMANCE ISSUES AND RELATED ANXIETY.

**FOR RELIEF OF STRESS DUE TO SUBOPTIMAL PERFORMANCE
JBuilder IS PROVEN EFFECTIVE ACROSS THE APPLICATION LIFECYCLE.**

JBuilder® from Borland® relieves the stress of performance anxiety associated with Application Development Dysfunction. A powerful component of Software Delivery Optimization (SDO) from Borland, JBuilder rapidly improves* individual performance and productivity. But it doesn't stop there. Thanks to deep integration across the entire Application Lifecycle, JBuilder can help your entire team create better software, faster. Don't let performance issues keep you from success.

Borland®



ASK BORLAND IF SDO IS RIGHT FOR YOU. borland.com/jbuilder

* JBuilder is indicated for organizations that wish to attain real competitive advantage in the marketplace. Side effects include increased margins, greater customer satisfaction and improved efficiency.
© 2005 Borland Software Corporation. All rights reserved. All Borland brand names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. 23389



Ajit Sagar
Contributing Editor

On the Tenth Year of Java My Client Asked of Me...

If we consider JavaOne as the event when Java was born, then June 27–30, 2005, will mark its tenth birthweek. A lot has happened since the language that was trademarked with dancing dukes made its appearance into the world of computing. In its current incarnation, the Java platform is undoubtedly the backbone of distributed enterprise applications in today's IT.

At Infosys, I've come across several projects in which the Java platform was being adopted, upgraded, integrated with existing technologies, or replaced with an alternate technology. The choices are influenced by business drivers that push enterprise applications toward the next generation platform, which can service more customers with a host of new features that were not available earlier due to technical limitations or poor design. Here are the main trends that I've seen at our client bases.

1. *Distribute My Apps*

This is obviously the main reason for enterprises to adopt the Java platform. Enterprises want to move toward an *n*-tier model, decoupling their UI and data source from the business logic and enabling scalable deployment. The trends we have seen here are the migration of applications from mainframes or two-tier applications such as PowerBuilder to an open platform – Java/J2EE.

2. *Orient My Services*

SOA is one of the hottest buzzwords in the industry. The promise of bridging the gap between technology and business is what makes service-oriented architecture such an attractive proposition. Java is a natural fit for enabling service orientation of existing applications. Several enterprises are considering the move toward a service-oriented architecture and looking at the features of the Java platform as the enabler.

3. *Service-Enable My Web*

SOA is often confused with Web services, which are basically the most feasible (but not the only) choice for implement-

ing an SOA. However, as the obvious technology of choice for service-enabling applications, many clients are looking at adopting/enhancing their Java applications to expose business functionality via Web services. Java WSDP, Apache's WSIF, and feature sets provided by several vendors make these initiatives feasible.

4. *Orchestrate My Processes*

We've had discussions with several clients on the merits of adopting BPM and workflow. The market for BPM, discounting Microsoft's BizTalk, is made up of players from different origins, who have wrapped their offerings under the BPM umbrella. The base for all these vendors is the J2EE platform. In this case, clients usually have a messaging and Java base that they want to enhance with process orchestration. There are many initiatives that are driven by the new compliance rules such as those under Sarbanes-Oxley.

5. *Migrate My App Server*

Enterprises already deployed on a J2EE application server are looking at upgrading/migrating/replacing their application servers. The drivers include obsolescence, acquisitions, and cost. If you are at JavaOne and are interested in this particular trend, please attend my presentation on Wednesday at 8:00 p.m. – **BOF 9042: Who Moved My J2EE Platform**, which discusses app server migrations. The details are at my blog: <http://www.ajitsagar.javadevelopersjournal.com/read/1063063.htm>.

6. *Open My Source*

Cost and TCO have driven enterprises to seriously consider the adoption of the Java platform through the open source community. While large enterprises are still wary of betting their entire business on open source Java products, component and framework development within these enterprises is definitely adopting the open source route. In addition, the majority of Java vendors are leveraging open source as the base for their own products.

7. *Integrate My Messages*

A large fraction of clients that we service have their applications built on some messaging base. This is not typically JMS, but a pre-JMS MOM product. Several organizations are dealing with the design issues of leveraging their existing messaging infrastructure with a Java front end. At the same time, they want to explore the benefits of Enterprise Service Bus (ESB) offerings from Java vendors.

8. *Integrate My Enterprise*

EAI and portals – the back-end and front-end integration of enterprise applications is a prevalent issue within large enterprises with a variety of technologies. The Java platform offers several features such as the Portlet API (JSR 000168) and JCA for EAI integration to enable application integration.

9. *Scale Back My EJBS*

A trend that is evident in several large organizations is to scale back on the usage of EJBS. I don't know of many clients who are recommending entity beans for application design. In fact, the directive is usually to replace entity beans with in-house persistence techniques, hibernate, or JDO. Message-driven beans (MDBs), however, are being leveraged where appropriate.

10. *And Take Me Back to the Mainframe*

This is one of the counter-intuitive moves in some organizations. For number crunching and batch functionality, many clients have realized that embracing Java for everything was overkill. In many cases, the functionality is being moved back to where it was, with the distribution of the application being mainly in the UI. Leveraging Web services directly to integrate with legacy apps is another trend.

All in all, Java has come a long way in the past 10 years. Enterprises have come the full circle from wariness, to faith, to trust, to an optimal utilization of what the platform has to offer. ☺

Ajit Sagar is a senior technical architect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has been working with Java since 1997, and has more than 15 years experience in the IT industry. During this tenure, he has been a programmer, lead architect, director of engineering, and product manager for companies from 15 to 25,000 people in size. Ajit has served as JDJ's J2EE editor, was the founding editor of XML Journal, and has been a frequent speaker at SYS-CON's Web Services Edge series of conferences. He has published more than 75 articles.
ajitsagar@sys-con.com

Perforce SCM. Bring your local heroes together.



Perforce Software Configuration Management brings you and your team together - wherever you happen to work.

[Fast]

[Scalable]

[Distributed]

You've got developers in New York, London and Singapore collaborating on your project. Your UI designer is in San Francisco and your QA team is in Buenos Aires. SCM nightmare? No. Just another day's work for Perforce.

Perforce Proxy, part of Perforce's multi-site solution, brings files to remote users on demand and caches them locally for other users at the same location. Local caching guarantees quick response times for remote users while maintaining real-time access to project activity and status information.

Unlike replication solutions with their inherent administrative overhead, a Perforce multi-site solution recovers files automatically, reduces the load on the Perforce Server, and requires no remote backups.

Fully integrated into Perforce's client/server architecture, Perforce Proxy requires only TCP/IP connectivity. A single Perforce Server can support any number of Perforce Proxy sites, allowing developers all over the world to collaborate.

PERFORCE
SOFTWARE

Download a free copy of Perforce, no questions asked, from www.perforce.com. Free technical support is available throughout your evaluation.

All trademarks used herein are either the trademarks or registered trademarks of their respective owners.

Shouldn't J2EE Be More Like Java?

Building a better bean

by Kenneth D. Kruszka

The elegance of Java stems from how the language addresses a number of highly complex software engineering issues in a seemingly consistent and easy-to-use paradigm. While there are a few potholes that you need to be mindful of, most caused by the differences between primitives and objects, the power and reach of standard Java are testaments to the principle of simplicity that is embodied within it. Unfortunately, when it comes to Java's enterprise platform, J2EE strays considerably from J2SE in a number of areas, creating unnecessary additional complexity and ambiguity. To make matters worse, J2EE 1.3 (EJB 2.0) introduced self-inconsistencies.

J2EE's deviations fall into two categories: Structural and Behavioral. Structural deviations concern those areas where the implementation of a J2EE component requires a format that differs significantly from the implementation format of a similar J2SE component. Behavioral deviations concern those areas where substantial inconsistencies exist in the fundamental workings of J2EE, be they self-inconsistencies in J2EE components or inconsistencies with respect to analogous J2SE components.

In the case of J2EE's most high-profile technology, Enterprise JavaBeans (EJBs), these digressions can be mitigated by extending and enhancing EJBs into what I'll term BetterEJBs (see the pullout at the end of this article). Just as application server deployment tools can transparently generate EJB stubs and skeletons, tools can likewise be developed to auto-generate the necessary artifacts that are derivable from the BetterBean. The benefit: a BetterEJB developer needs only implement a single class, cutting down on development time, complexity, and the potential for errors while increasing understandability through the consolidation of information in a single location.

This article discusses a number of issues regarding the disparity between J2EE and J2SE, and spells out practical approaches and techniques that can be used to help return J2EE, specifically EJB, back to its Java roots. Additionally, this article

illustrates the derivation of all supporting artifacts from the BetterBean, suggesting toolable functionality that could be realized to auto-generate the boilerplate constituent parts of the BetterEJB.

Structural Deviations

J2EE, and specifically EJB, has complicated the development of reusable functionality. Besides compromising Java's object-oriented nature, the format of the code has been made needlessly complex. When a data value object or utility object can be written in a single Java class, why does J2EE require that *four* (or possibly even *six*) files be written to make the enterprise-grade equivalent: an Entity or Session EJB? An EJB requires not only the Bean class, but also a Home Interface, a Remote Interface (optionally, a Local Home Interface and Local Remote Interface), and a deployment descriptor.

Given the distributed nature of EJB deployment, which is realized through application server-generated Stubs and Skeletons, it's reasonable for enterprise-grade functionality to require a client-facing interface in addition to the bean class. But, the necessity of any additional artifacts is overly complex and error prone.

Declarative Programming Gone Awry

By far the most egregious deviation is the EJB deployment descriptor. While declarative programming, through the use of runtime configuration files, is useful for separating environment-dependent information from hard-coded, core-component functionality, EJB has mistakenly taken declarative programming too far. Is it not central to the operation of a session bean whether it's stateful or stateless? It's not fundamental to the operation of a method whether or not it uses a database transaction. Considering that details such as these are integral to the functioning of the code, they shouldn't be alterable after compilation; these details should be explicit in the code and shouldn't be tweaked in deployment descriptors.

The following code illustrates how the designation of a BetterEJB as stateless or

stateful is accomplished through the use of a marker interface in the definition of the BetterBean class, BetterBeanA:

```

1 public class BetterBeanA
2     implements StatelessBetterBean
3 {
4     ...
5     public Boolean foo()
6     {
7         return Boolean.TRUE;
8     }
9 }
```

Because BetterBeanA implements the StatelessBetterBean marker interface, the derived deployment descriptor is for a <session> bean with <session-type> of Stateless, as shown in the following snippet:

```

1 <session>
2 <ejb-name>
3     BetterBeanA
4 </ejb-name>
5 <home>BetterHomeInterfaceA</home>
6 <remote>
7     BetterRemoteInterfaceA
8 </remote>
9 <local-home>
10    BetterLocalHomeInterfaceA
11 </local-home>
12 <local>
13    BetterRemoteInterfaceA
14 </local>
15 <ejb-class>BetterStubA</ejb-class>
16 <session-type>
17     Stateless
18 </session-type>
19 </session>
```

The benefit of using marker interfaces is that it becomes possible to do build/compile-time checks on BetterBeanA, which can help eliminate the possibility of difficult-to-find runtime bugs or cryptic deployment-time errors.

Unnecessary Multitude of Interfaces

Why do EJBs require both a HomeInterface and a RemoteInterface (and optionally a LocalHomeInterface and LocalRemoteInterface)? Yes, the Ho-



Ken Kruszka is a product manager and architect who has developed software solutions for NASA, US Department of Defense, NIST, the telecommunications industry, and the financial services industry. Ken holds a BS and MS in computer science, and an MBA.

kdkruszka@yahoo.com

The Developer Paradox:

No time to test your code? But spending **long hours** reworking it & resolving errors?



Check out **Parasoft Jtest® 7.0**
Automates Java testing and code analysis.
Lets you get your time back and deliver quality code with less effort.

■ **Automated:**

Automatically analyzes your code, applying over 500+ industry standard Java coding best practices that identify code constructs affecting performance, reliability and security.

Automatically generates and executes JUnit test cases, exposing unexpected exceptions, boundary condition errors and memory leaks and evaluating your code's behavior.

Groundbreaking test case “sniffer” automatically generates functional unit test cases by monitoring a running application and creating a full suite of test cases that serve as a “functional snapshot” against which new code changes can be tested.

■ **Extendable:**

Industry standard JUnit test case output make test cases portable, extendable and reusable.

Graphical test case and object editors allow test cases to be easily extended to increase coverage or create custom test cases for verification of specific functionality.

■ **Integrated:**

Integrates seamlessly into development IDE's to support “test as you go” development, and ties into source control and build processes for full team development support.

To learn more about Parasoft Jtest or try it out, go to www.parasoft.com/Jtest



Automated Software Error Prevention

meInterface is used for locating the EJB implementation, while the RemoteInterface is the interface that's realized by the EJB implementation.

But, is it not possible for a single interface to perform both functions? Such a combination is achieved by the BetterInterface, as is shown by the interface for BetterBeanA, BetterInterfaceA:

```

1 public interface BetterInterfaceA
2 {
3     public static final String[2]
4         JNDI_LOOKUP_REFS
5         = {"BetterBeanA/LocalHome",
6           "BetterBeanA/Home"};
7     ...
8     public Boolean foo();
9 }

```

Listing 1: Using Pass-by-value-return to simulate Pass-by-handle

```

/* BetterStub example
 * reconcile() implemented in base class
 */
/* BetterBean example
 */

1 public class BetterStubB
2     extends BetterStubBase
3 {
4     public Boolean bar(Set set)
5     {
6         ArrayList values = new ArrayList();
7         List remoteValues;
8         values.add(set);
9         // lookup home and remote interface
10        ...
11        remoteValues = remoteB.callBar(
12            values);
13        Boolean result = (Boolean)
14            reconcile(
15                values, remoteValues);
16        return result;
17    }
18 }

/* BetterSkeleton example
 */
1 public class BetterSkeletonB
2     implements SessionBean,
3         BetterSkeleton
4 {
5     public List callBar(
6         ArrayList values)
7     {
8         Set param0 = (Set)values.get(0);
9         Object result;
10        BetterBeanB bean
11            = new BetterBeanB();
12
13        result = bean.bar(param0);
14        values.add(result);
15        return values;
16    }
17 }

1 public class BetterBeanB
2     extends BetterBeanBase
3     implements StatelessBetterBean
4 {
5     public Boolean bar(Set set)
6     {
7         set.add("bar");
8         return Boolean.TRUE;
9     }
10 }

```

In addition to providing client-side signatures for all the methods implemented by BetterBeanA, this interface holds references for both remote EJB lookup and local EJB lookup. Passing these references to a lookup utility function (described later in this article) allows for retrieving either reference. In this way, only one interface is needed for both the lookup of and interaction with a BetterEJB.

Behavioral Deviations

While structural deviations concern the format of the artifacts associated with EJBs, behavioral deviations concern the fundamental workings of EJBs. Behavior deviations arise from inconsistencies that J2EE introduced either with respect to analogous J2SE components or within J2EE itself. They are often side effects caused by the "Enterprise Services" that application servers are required to provide, such as EJB location transparency.

EJBs Break Pass-By-Handle

Java employs a parameter passing mechanism for objects that has come to be known as pass-by-handle. Technically, Java obeys strict pass-by-value semantics, which means that the value assigned to a method parameter is a copy of the argument value. Paraphrasing Bruce Eckel's explanation in *Thinking in Java*: when member fields of parameter objects are modified within the called method, those changes alter the argument object held by the caller and persist after the called method ends. The reason for this is that object handles are passed-by-value when making method calls, not the objects themselves.

However, EJB method invocations don't follow this established pass-by-handle behavior. Instead, object arguments to remote methods are deep-cloned, through serialization. As a result, modifications to parameter objects aren't reflected in the argument objects held by the caller.

To further complicate matters, EJB 2.0 opened a rift in EJB behavior with the introduction of Local interfaces. As described on page 148 in *Enterprise JavaBeans* (3rd edition) by Richard Monson-Haefel, "The Local Client API also passes object arguments by [handle] from one bean to another... This means that an object passed from enterprise bean A to enterprise bean B is referenced by both beans, so if B changes its values A will see those changes."

To put it succinctly, pass-by-handle semantics are in place when using Local interfaces, but not Remote interfaces. The advice given by Monson-Haefel to ensure consistent behavior on EJB method calls is to always pass immutable objects or copies of mutable objects when making method calls on Local EJBs [page 149]. But I respectfully submit that this advice misses the mark entirely. The problem is not that the parameter-passing semantics of Local EJBs differ from those of Remote EJBs, but that the semantics of Remote EJB calls differ from those of ordinary Java calls. We should be seeking ways to make Remote EJBs behave more like Local ones, not the other way around.

The BetterEJB addresses this discrepancy by simulating the pass-by-handle mechanism in EJB method calls; BetterStub and BetterSkeleton work together to bridge the chasm of remote method invocation through the revival of pass-by-value-result semantics, which first appeared in Algol W but are foreign to most modern programming languages.

Pass-by-value-result, as realized in the code in Listing 1, works in the following way: In the BetterStub method, the parameter objects are incorporated into a List that is passed as a single argument to the remote method on the BetterSkeleton. The BetterSkeleton extracts the objects from the List and passes them as individual arguments to the BetterBean method. The BetterBean may alter the contents of the parameters thereby modifying the objects held by the BetterSkeleton. On the return, the BetterSkeleton adds the BetterBean return value to the List and returns the full List back to the BetterStub, which then reconciles its local objects with the possibly modified objects in the return list. In this way, the client-side objects are updated with any changes that were made on the remote side of the EJB call.

The benefit of this approach is that BetterEJB method calls work in a manner similar to ordinary Java method calls; if parameter objects are modified, those side effects are seen in the caller's version of the objects. The shortcoming of this approach is that changes made to parameter objects in the BetterBean aren't propagated instantaneously; the caller's version of the objects isn't altered until control is returned from the remote method call.

Innovations by InterSystems



Real-Time Data Analytics With A Real-Fast Database.

Imagine being able to query a lightning-fast operational database in real time.

Now you can, with our multidimensional database for transaction processing and real-time analytics.

Only Caché combines robust objects and robust SQL, thus eliminating object-relational mapping. It requires little administration, delivers speed and scalability on minimal hardware, and comes with a rapid application development environment.

These innovations mean faster time-to-market, lower cost of operations, and higher application performance. We back these claims with this money-back guarantee: *Buy Caché for new application development, and for up to one year you can return your license for a full refund if you are unhappy for any reason.**

Innovative database. Guaranteed performance.

InterSystems
CACHÉ[™]



Rapid Integration Platform Makes Applications Perform Together.

Imagine being able to get your applications to perform together as an ensemble. Easily.

Now you can, with our universal integration platform.

Ensemble is the first fusion of an integration server, data server, application server, and portal development software – in a single, seamless product. This is the complete ensemble of technologies needed for rapid integration, fast development, and easy management.

These innovations mean all of your integration projects will be completed on time and on budget, with a simplified learning curve for your IT staff. We back these claims with this money-back guarantee: *For up to one year after you purchase Ensemble, if you are unhappy for any reason, we'll refund 100% of your license fee.**

Innovative integration. Guaranteed performance.

InterSystems
ENSEMBLE[™]

For a free copy of CACHÉ, or to request a free ENSEMBLE proof-of-concept project, visit www.InterSystems.com/Free8P

*Read about our money-back guarantees at the web page shown above.

© 2005 InterSystems Corporation. All rights reserved. InterSystems Caché and InterSystems Ensemble are trademarks of InterSystems Corporation. 5-05 ComboImno8JaDeJo

An alternate solution to this issue would be to always pass EJB-wrapped value objects as parameters. Then, the value objects would be location-transparent and there would only be a single copy with which all clients interact. However, the performance degradation of such an approach is prohibitive due to the overhead associated with the remote calls when accessing any attributes on the value objects.

Local Interfaces Prevent Load Balancing and Location Transparency

EJB was a gigantic step forward for Java. The introduction of EJB eased the development of distributed applications in Java. And there was much rejoicing, until developers discovered the performance degradation that came with the assumptions intrinsic to EJB, especially those concerning

location transparency. In response to these concerns, EJB 2.0 introduced Local Interfaces, which allowed for calls between collocated EJBs to sidestep the network overhead inherent in remote method calls.

While the intention behind Local Interfaces was laudable, the implementation effectively gutted core application-server services, most notably load balancing and location transparency, and put the onus back onto application developers. The introduction of Local Interfaces forced developers to program explicitly in their client code through which interface they would interact with the EJB: Local or Remote. (Given the difference in the parameter-passing semantics outlined above, it's probably a good thing that EJB 2.0 required developers to explicitly code for this. But, I digress.)

To restore centralized load balancing and the notion of location transparency of EJBs, an intelligent EJB lookup mechanism is needed, such as the one shown in Listing 2.

This is a simple greedy approach to load balancing. If the local server isn't overtaxed, then use an EJB deployed to this server, otherwise use a remote EJB. Considering the overhead involved with remote method calls, this is a reasonable approach.

The shortcoming of this approach is the difficulty in developing load-balancing heuristics. But the benefit of this approach is that this functionality, which restores services that should be done by the application server, can be removed from the purview of application developers, and provided as a centralized factory method. This lets developers use any BetterEJB without having to know where it is deployed, and enables changes in deployment architecture without requiring code modifications.

EJB 3.0 to the Rescue?

The promise of EJB 3.0 is to simplify EJB development dramatically, and the EJB 3.0 expert group has taken giant strides toward that goal. Among the improvements in EJB 3.0 (many of which are beyond the scope of this article) is the elimination of Home interfaces, reducing the number of artifacts that must be created by developers. Furthermore, EJB 3.0 leverages annotations as

a way to provide declarative metadata interspersed in class code, thereby eliminating the need for separate deployment descriptors.

However, EJB 3.0 doesn't go far enough. While some structural deviations are lessened, they aren't completely resolved. The use of annotations does mitigate the need for deployment descriptors, but annotations are still a form of declarative programming. In many cases this declarative programming paradigm is unnecessary; instead of annotations, simple marker interfaces would work just as well. Worse still, EJB 3.0 fails to address behavioral deviations at all.

Conclusion

J2EE is supposed to be an enterprise extension of Java. As such J2EE should behave like J2SE, but in a distributed environment, without requiring unnecessary additional effort on the part of developers. Applications deployed in a distributed environment need to take into account a number of key architectural issues including: failover, location transparency, and load balancing. Aren't these some of the reasons why we're buying application servers? The application servers should take care of such issues for us; J2EE should mandate that such issues be taken care of for us. But, this isn't the case now. Until the J2EE expert groups bridge the gap back to Java, both structurally and behaviorally, the only solution we have as application developers and architects is to address these issues ourselves. (See *The Better EJB* sidebar on page 18). ☺

Resources

- Burke, B. "EJB 3.0 Preview. Part 1: The basic programming model." <http://www.sys-con.com/story/?storyid=46975&DE=1>
- Burke, B. "EJB 3.0 Preview. The advanced features Part 2." <http://www.sys-con.com/story/?storyid=47351&DE=1>
- Eckel, B. (1998). *Thinking in Java*. Prentice Hall PTR.
- Gamma, E., et al. (1995). *Design Patterns*. Addison-Wesley.
- Monson-Haefel, R. (2001). *Enterprise Java Beans*. Third Edition. O'Reilly.
- Proulx, E. "EJB Inheritance." <http://www.onjava.com/pub/a/onjava/2002/11/13/ejbinherit3.html>

Listing 2: Load-balanced BetterEJB lookup

```

/* BetterEJB lookup utility snippet
 */
1 private static final Class[]
2   NO_PARAMETER_TYPES = new Class[0];
3
4 private static final Object[]
5   NO_PARAMETER_OBJECTS = new Object[0];
6
7 public BetterInterface
8   lookupBetterEJB(
9     Context jndiContext,
10    String[] ejbRefs)
11   throws Exception
12   {
13     EJBLocalHome   ejbLocalHome;
14     EJBHome        ejbHome;
15     BetterRemote   betterRemote;
16     Class          ejbClass;
17     Method         createMethod;
18     BetterStub     stub;
19
20     if (serverLoadIsLow())
21     {
22       ejbLocalHome
23         = (EJBLocalHome)context.lookup(
24           ejbRef[0]); // local ref
25       ejbClass = ejbLocalHome.getClass();
26       createMethod
27         = ejbClass.getMethod(
28           "create",
29           NO_PARAMETER_TYPES);
30       betterRemote = (BetterRemote)
31         createMethod.invoke(
32           ejbLocalHome,
33           NO_PARAMETER_OBJECTS);
34     }
35
36     if (betterRemote == null)
37     {
38       ejbHome
39         = (EJBHome)context.lookup(
40           ejbRef[1]); // remote ref
41       ejbClass = ejbHome.getClass();
42       createMethod = ejbClass.getMethod(
43         "create",
44         NO_PARAMETER_TYPES);
45       betterRemote = (BetterRemote)
46         createMethod.invoke(
47           ejbHome,
48           NO_PARAMETER_OBJECTS);
49     }
50     stub = betterRemote.getBetterStub();
51     return stub;
52   }

```

Your potential. Our passion.™
Microsoft

Think big. Then build.

With Visual Studio®.NET 2003, you can build enterprise Web applications with less code, so you can turn that big idea into reality faster than you ever thought possible. For example, RAD-style Web Forms let you quickly build applications for any browser or platform. Plus, the enhanced HTML editor gives you IntelliSense® statement completion for HTML tags. All of which means you're more productive, and more ready to take on your biggest ideas. Find out more at msdn.microsoft.com/visual

© 2004 Microsoft Corporation. All rights reserved. Microsoft, IntelliSense, Visual Studio, the Visual Studio logo, and "Your potential. Our passion." are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

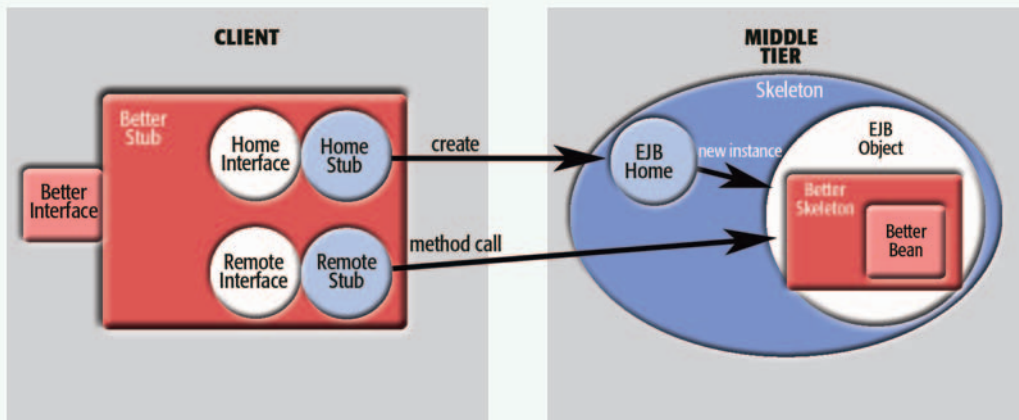


Microsoft
Visual Studio

The Better EJB

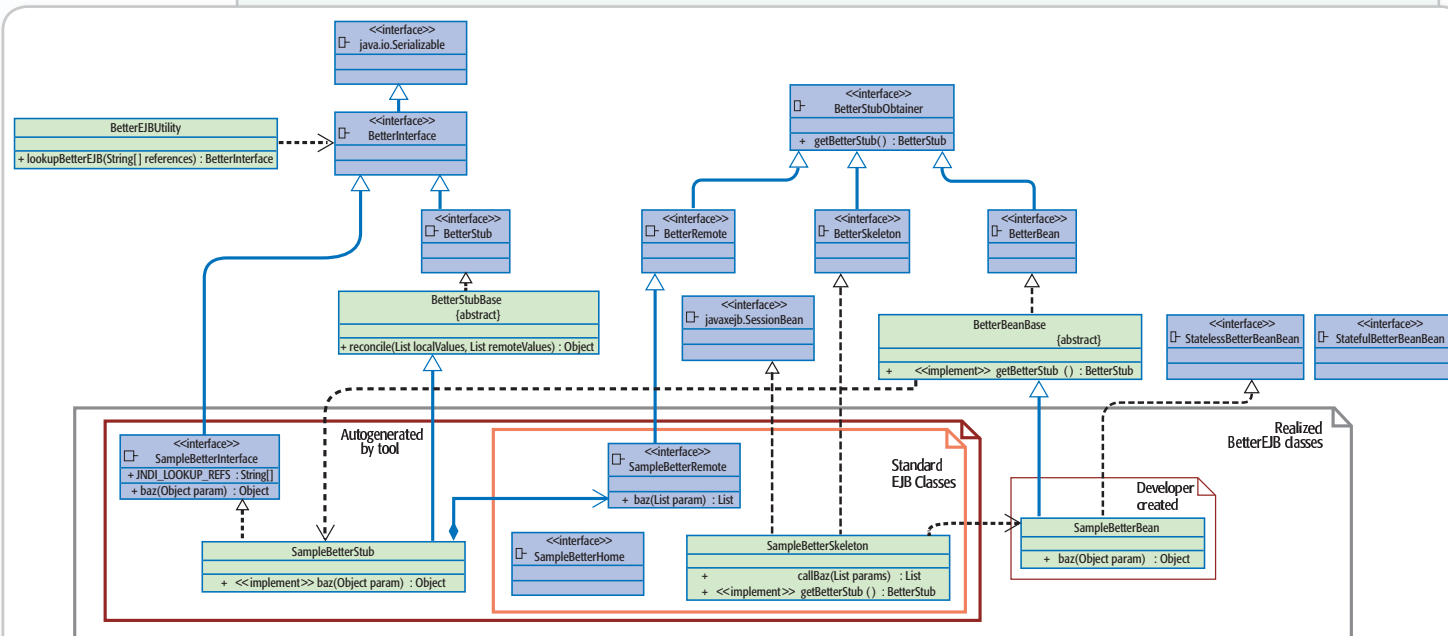
The architecture diagram illustrates how BetterEJB components (rectangles) leverage ordinary EJB components (ovals/circles). Just as application servers auto-generate service-providing classes (blue) from developer created classes and interfaces (white), so too can tools auto-generate code (red) from the developer created BetterBean (light red) to provide the higher-level services of BetterEJB.

- **BetterInterface** is a single interface for the Better EJB. No more fooling around with the Home, Remote, Local Home, Local Remote interfaces.
- **BetterStub** implements BetterInterface and lives on the client side of the call, working together with BetterSkeleton to provide the enhanced services of BetterEJB.
- **BetterSkeleton** lives on the middle tier side of the call, working together with BetterStub to provide the enhanced services of BetterEJB. Acts as a proxy to the BetterBean.
- **BetterBean** implements either StatelessBetterBean or StatefulBetterBean and contains the logic of the BetterEJB.



BetterEJB Architecture

The heart of the BetterEJB is the BetterBean class, which is an ordinary Java class, allowing for true object-oriented development. Like the EJB stub and skeleton, BetterStub and BetterSkeleton work together to handle the bookkeeping required for remote method invocation. It's this additional bookkeeping that helps bring J2EE back home to Java.



Sample BetterEJB Class Diagram

Think Crystal is a good fit for your Java application? Think again.

Visit
JReport
at
JavaOne!



Think JReport.

Forcing a Windows reporting solution into a J2EE environment is never going to result in a perfect fit. Only JReport is built from the ground up to leverage J2EE standards and modular components for seamless embedding in any Web application.

JReport is ready out-of-the-box, with all the tools necessary to empower your application for any reporting requirement. From reusable, shared report components to flexible APIs, JReport is the most complete embedded reporting solution available.

With the ability to scale to multi-CPU and server clusters, JReport is a perfect fit for any reporting workload. Load balancing and failover protection provide peak performance and uninterrupted access to critical business data. In addition, JReport integrates with any external security scheme for single sign-on.

JReport's ad hoc reporting lets users access and analyze data on demand, from any browser. And, with cascading parameters, embedded web controls for dynamic sorting and filtering, drill-down and pivot, JReport ensures users get the information they want, when they want it, and how they want it.

See for yourself why over half of the world's largest organizations have turned to JReport to enable their J2EE applications with actionable reporting.

When it comes to embedded Java reporting, JReport is the perfect fit. Download a FREE copy of JReport today at www.jinfony.com/jp6.

 **JReport**
JINFONET SOFTWARE

Java Application Security in the Corporate World

Java security isn't a skill of Java architects



by Adam Kolawa and Yakov Fain

The vast majority of corporate developers truly believe that application security is not their concern, assuming that network and engineering groups will build their environment in a secure way. But what about application security? Are you ready for the code audit?



Dr. Adam Kolawa is cofounder and CEO of Parasoft, a leading provider of Automated Error Prevention software solutions.

Dr. Kolawa, co-author of *Bulletproofing Web Applications* (Hungry Minds, 2001), has contributed to and written over 100 commentary pieces and technical articles for publications such as *The Wall Street Journal*, and *IEEE Computer*. He has also authored numerous scientific papers on physics and parallel processing. His recent media engagements include CNN, CNBC, the BBC, and NPR. Dr. Kolawa holds a PhD in theoretical physics from the California Institute of Technology, and has been granted 10 patents for his recent inventions. He is a well-known writer and speaker on industry issues and in 2001 was awarded the Los Angeles Ernst & Young Entrepreneur of the Year Award in the software category.

ak@parasoft.com

Application Security Isn't Getting the Attention It Deserves

When most people in the corporate world talk about “security,” they mean the security of the network, operating system, and servers. Organizations that want to protect their systems against hacker attacks invest a lot of time, effort, and money ensuring that these three components are secure. Without this secure foundation, systems cannot operate securely.

However, even if the network, server, and operating system are 100% secure, vulnerabilities in the application itself make a system just as prone to dangerous attacks as unprotected networks, operating systems, and servers would. In fact, if an application has security vulnerabilities, it can allow an attacker to access privileged data, delete critical data, and even break into the system and operate at the same priority level as the application, which is essentially giving the attacker the power to destroy the entire system. Consequently, the security of the application is even more important than the security of the system on which it's running. Building an insecure application on top of a secure network, OS, and server is akin to building an elaborate fortress, but leaving the main entryway wide open and unguarded.

There is a simple explanation to why this happens: tight project deadlines and unawareness of potential consequences. Project managers believe that answering that annoying review of the corporate security group takes care of everything. Not every project is reviewed by experienced enterprise architects, and even if it is, Java security is not one of the major skills of Java architects.

Most Developers Don't Know How To Write Secure Code

Most developers have no idea what writing secure code involves. Most have never thought about writing secure code

- probably in response to the corporate world virtually ignoring application security, and very few have ever had to try writing secure code. Some developers have heard that buffer overflows and SQL injections can cause security problems, but that's about the extent of most developers' security knowledge.

When developers are asked to make applications secure, they start trying to find security bugs in the application — after it's been built. For example, they might look for dangerous method calls and remove them, using an application vulnerability scanner, or using a security mechanism such as `mod_security` or an application firewall to prevent exploitation. However, this bug-finding strategy isn't sufficient to meet today's complex security requirements, such as those mandated by the Sarbanes-Oxley Act. Testing problems out of the application is both inefficient and largely ineffective. Independent, end-of-process bug finding alone can't and on't expose all possible security vulnerabilities.

With penetration testing, which involves trying to mimic an attacker's actions and checking if any tested scenarios result in security breaches, security vulnerabilities will go unnoticed unless the tester has the skill and luck to design the precise attack scenarios required to expose them. Considering that there are thousands, if not millions, of possible scenarios for even a basic application, odds are some vulnerabilities will be overlooked. However, it takes only one security vulnerability to compromise the security of an application and its related systems — opening the door to attacks, as well as fines for not complying with security mandates.

Furthermore, penetration testing can fail to catch the most dangerous types of problems. Let's assume that you have a Web application to test, and this application has a backdoor that gives admin privileges to anyone who knows to supply a secret argument, like `h4x0rzRgr8 = true`. A typical penetration test against a Web application uses known exploits and sends modified requests to exploit common coding problems. It would take years for this test to find this kind of vulnerability through penetration testing. Even an expert security analyst would have a tough time trying to exploit this. What about a difficult-to-reach section of code in the error-handling routine that performs an unsafe

database query? Or the lack of an effective audit trail for monitoring security functions? These kinds of problems are often entirely overlooked by even a diligent penetration test.

Other popular end-of-process security testing techniques - such as using static analysis to check whether code follows a standard set of security rules such as “Do not use java.util.Random” or “Use java.security.SecureRandom” - might expose some of the vulnerabilities that penetration testing overlooks, but come with their own share of problems. For instance, consider some of the weaknesses of trying to identify security vulnerabilities through static analysis. One is that these patterns don’t consider the nuances of actual operation; they don’t factor in business rules, or general security principles. If you have a Web application that lets your customer see their competitor’s account by adding one to the session ID, this is a very serious problem. However, this kind of problem escapes static analysis because it doesn’t involve a dangerous function call. Security assessment, in this sense, isn’t always a bug to find, but a design problem to verify. Another problem is false positives. Static analysis can’t actually exploit vulnerabilities; it can only report potential problems. Consequently, the developer or tester must review every reported error and then determine if it indicates a true problem, or a false positive. Sophisticated static analysis methods can improve accuracy, but ultimately, a significant amount of time and resources must be spent reviewing and investigating reported problems and determining which actually need to be corrected.

Complying with Sarbanes-Oxley

To comply with Sarbanes-Oxley (SOX), public companies need to effectively define and verify security policies for their financial and record-keeping applications

Public companies are now required by SOX to implement and verify effective security for their financial and record-keeping applications. To comply with this requirement, it’s necessary to establish an effective application security policy and verify that the policy is actually implemented in the code and reflected in the system functionality. By security policy we mean a document that defines best practice secure coding standards, secure application design rules, security testing benchmarks, privacy requirements, as well as custom security requirements.

According to SOX, having a security policy has evolved from a “nice-to-have” feature to an essential business requirement. Companies that don’t establish and implement effective security policies could now be found to be negligent and face significant fines for failing to comply with SOX. A lot of developers and managers still treat security like they treat quality — they try to get as much quality/security as they can to the best of their knowledge, but often settle short of complete quality/security. However, systems that aren’t 100% secure aren’t acceptable under SOX. If development managers don’t recognize this, they could cause their companies tremendous liabilities .

Defining a security policy doesn’t satisfy SOX requirements; the specification items defined in the policy must actually be implemented in the code. In other words, the specification must truly be seen as requirements — not as suggestions or guidelines, as is typically the case with functionality specifications. The specifications defined in

<p>Use JAAS in a single, centralized authentication mechanism. Instead of doing access control checking in each servlet or JSP, use a “front-door” servlet to do the access control checking.</p>	<p>The following code is not allowed:</p> <pre>class ISOGenericServlet extends Servlet { void doPost(HttpServletRequest req, HttpServletResponse resp) { lc.login(); } }</pre> <p><i>JAAS Authentication cannot take place outside of the ISOAuthentication.</i></p> <p>The following code must be implemented:</p> <pre>class ISOAuthentication { void authenticate(String user, String pass) throw LoginException { //do authentication } }</pre> <p><i>This code is called every time a JAAS authorization operation takes place</i></p>
<p>Do not cause deadlocks by calling a “synchronized” method from a “synchronized” method.</p> <p>Avoid potential deadlock conditions that can cause denial of service.</p>	<p>The following code is not allowed:</p> <pre>public synchronized void method1() { method2(); }</pre> <pre>public synchronized void method2() { }</pre> <p><i>Synchronized methods that call other synchronized methods are dead lock prone. Try to write the code so that a thread doesn't try to get a lock on a monitor while already holding a lock. One possibility is to use a “synchronized” statement to only synchronize the part of the method that really needs to be synchronized. Alternatively, before locking a second object, ensure that it is not already locked.</i></p>
<p>Use only strong cryptographic algorithms.</p> <p>Some encryption algorithms are not as strong and therefore not as secure as others. It is recommended that the strongest practical cryptography be used and weak cryptographic algorithms avoided</p>	<p>The following code is not allowed:</p> <pre>Cipher.getInstance("MD5");</pre> <p><i>Do not use cryptographic algorithms that have known problems that defeat the effectiveness of the cryptography.</i></p> <p>The following code shows the correct behavior:</p> <pre>Cipher.getInstance("SHA-1");</pre> <p><i>Only use strong cryptographic algorithms.</i></p>
<p>Validate ‘HttpServletRequest’ object when extracting data from it.</p> <p>HttpServletRequest objects contain user-modifiable data that, if left unvalidated and passed to sensitive methods, could cause serious security problems such as SQL injection and cross-site scripting (XSS).</p>	<p>The following code is not allowed:</p> <pre>String name = req.getParameter("name");</pre> <p><i>Unvalidated user data could be passed on to sensitive methods.</i></p> <p>SQL Injection occurs when user input (for example, from HttpServletRequest) is appended to a SQL query and passed to a database without being properly validated. An attacker can exploit this vulnerability to access and modify privileged data and execute arbitrary commands. Cross-site scripting problems occur when user-modifiable data is output verbatim to HTML. Subsequently, an attacker can submit script tags with malicious code that is then executed on the client browser. This allows an attacker to deface a site, steal credentials of legitimate users, and gain access to private data.</p>
<p>Session tokens should expire.</p>	<p>The following code shows the correct behavior:</p> <pre>long sessionAge = System.currentTimeMillis() - session. getCreationTime(); if (sessionAge > maxAge) { session.invalidate(); }</pre>

Table 1 Sample Security Policy Enforcement

“Application security is one of multiple issues that outsourcing brings to the corporate table”

the security policy *must* be implemented...no ifs, ands, or buts. If your corporate information group doesn't have resources to enforce this, your architecture group may have to take this responsibility.

What's required to ensure that the security policy is implemented in the code? First, code should be statically analyzed to enforce the organization's security policy on the client and server sides. Static analysis typically looks for potentially dangerous function call patterns and tries to infer if they represent security vulnerabilities (for instance, to determine if code has unvalidated inputs, and if unvalidated inputs are passed to specific functions that can be vulnerable to attack).

Next, thorough automated penetration testing should be done to confirm that the security policy has been implemented correctly and operates properly. In addition, security should be verified through unit testing, runtime error detection, and SQL monitoring.

Just scanning the code for known security bug patterns and performing some penetration testing isn't enough. You need to have a security policy that defines how the code should be built to safeguard security, as well as how the code should be tested to verify that the required security was implemented.

Security Policy

What does a security policy involve? First, you define how the code needs to be written so that it isn't vulnerable to attack. This policy should be designed to prevent both types of possible security bugs: bugs in the code that cause security mechanisms to malfunction, and security mechanisms that aren't implemented correctly. The first case tends to be a problem when critical security tasks such as input validation or authentication are handled differently in different parts of the code. Not only is this bad for maintainability, it's bad for security because it introduces more attack surfaces where vulnerabilities can hide.

When implemented, all security-related operations specified in the security policy should be concentrated in one segment of the application. You can then focus your resources on verifying and maintaining the security of that one critical module. This centralized security policy acts like a drawbridge for a castle: it isolates the area attackers can exploit and allows for a more focused defensive strategy.

Table 1 shows excerpts from a security policy for a Java-based application.

Outsourcing and Security

Application security is one of multiple issues that outsourcing brings to the corporate table. For example, can you allow developers in other countries to have access to such sensitive information as social security numbers and bank

account numbers? In developing countries the chances of such information being stolen are higher. This introduces the additional expense of creating separate environments for such teams (installing separate database and J2EE servers, and deploying data-scrambling software).

If you are outsourcing support of your applications, have you arranged for auditing the administrator's actions? If a user has been granted access to particular screens or specific data, do you have a record of who did it and when?

In some cases companies even outsource the process of running penetration tests.

Summary

The main goal of this article was to bring your attention to potential issues and security holes in your applications. Set and enforce security policies in your organization and consider doing penetration tests and static analysis of Java code using automated software testing tools. ☛

Sarbanes-Oxley and Information Technology

Sarbanes-Oxley Act was signed into law by President Bush in July of 2002. It requires public companies to improve the accuracy and reliability of corporate reports and disclosures to prevent and punish corporate fraud. It has provisions for auditor independence and corporate responsibilities and sets stringent standards for corporate executives. This act was named after Senator Paul Sarbanes and Representative Michael G. Oxley.

One section of the law says that financial reports must be accurate and have to be certified by a company's top executives on a quarterly basis. From an IT point-of-view, this not only means that the software that produces such reports must be accurate, but also that it must be secure enough to prevent attempts to modify reports during or after their creation. Another section forces corporations to set effective internal control for reporting. Among other inspections, independent auditors can check if the application software keeps track of the deletion or modification of sensitive data.

This law requires that changes in the financial state of a corporation must be made available to the public in a timely manner. For IT this means that the infrastructure must include disaster recovery sites and data replication procedures that ensure the availability of such information to the public even if the primary data center is down.

For more details you can refer to the document "IT Control Objectives for Sarbanes-Oxley" published online by the IT Governance Institute.

As you can guess, corporate executives don't really like this law. They now need to spend a substantial part of their revenues on complying with the Sarbanes-Oxley Act.

They also need to pay more attention to the software quality and security or else they may face punishments anywhere from losing their job to jail sentences. They also have to think twice before saying "I do" to their partner outsourcers from overseas.

From the IT perspective, this law generates more jobs and new projects, especially in compliance departments. This act may not be as big as the Y2K hype, but it will definitely bring more people to the IT industry.



Yakov Fain is a Java 2 Enthusiast and Educator from Wall Street. He is the author of the best selling book, *The Java Tutorial for the Real World*, an e-book *Java Programming for Kids, Parents and Grandparents*. Fain also authored several chapters for *Java 2 Enterprise Edition 1.4 Bible*. On Sundays Yakov teaches Java (see www.smartdataprocessing.com)

yakovfain@sys-con.com

Pure *Java*
Pure *Excel*
Power for users
Control for IT
Pure *Paradise*

Formula One e.Spreadsheet Engine

*API-driven, embedded, 100% pure-Java,
scalable spreadsheet toolset*

Spreadsheets play an essential role in the business world. And now, they can also perform a vital function in your Java applications. How? With the Formula One e.Spreadsheet Engine, an API-driven, 100% pure Java toolset for embedding Excel spreadsheet functionality in your applications.

Excel-enable your Java applications

Use a state-of-the-art graphical development environment to build fully-formatted Excel report templates that include formulas, functions, colors, merged cells, even charts. It's fast, easy and effective.

Embed live, Excel-compatible data grids

Include interactive Excel forms and reports in your Java applications. Let users manipulate them using their spreadsheet skills and then commit the changes to databases or applications – or save them as an XLS file on their desktops.

Automate complex calculations and rules

Embed Excel spreadsheets that automate complex calculations and business rules on J2EE servers. Stop translating spreadsheet logic into Java code today, and start leveraging the development skills of your organization's spreadsheet experts.

Read and write server-based Excel spreadsheets

Give your users server-based spreadsheets populated with up-to-the-minute information directly from data-bases, XML files and enterprise applications. Get control of runaway data warehouses and spreadmarts now.

Make spreadsheets a part of your strategies

Visit us today at www.reportingengines.com to request a **free trial** of the e.Spreadsheet Engine. We'll show you how to make Excel spreadsheets a vital and productive part of your enterprise computing strategies.

ACTUATE
ReportingEngines

www.reportingengines.com
sales@reportingengines.com
888-884-8665 + 1-913-851-2200

**FREE TRIALS,
DEMOS AND
SAMPLE CODE**

Isolating Concurrent Java Apps in a Virtual Machine

by Murali Kaundinya

Peaceful coexistence

It's not at all uncommon to see a server machine or even a desktop machine that runs the same or multiple applications each with its own Java Runtime Environment. In the server environment, aside from the scaling issues with garbage collection, the real motivation was for different applications to not be adversely affected by sharing application data and state within the same JVM. However, launching and running multiple processes, even on a server, comes with a price.

JVM researchers have long investigated the possibility of running multiple applications concurrently on the same VM and yet have sufficient degree of isolation to not step on each other's toes. Java Isolates (JSR-121) is a JCP effort to come up with a standard set of APIs that when implemented in the platform can someday provide the degree of isolation between applications that could run concurrently and coexist on the same physical JVM.

Introduction

About four years ago, a large client had experienced a sporadic outage with their application that was running on a J2EE server. There were multiple applications deployed on the same server, each having their own URL. The code review within these applications did not reveal anything untoward and they did not load any native libraries. The application server in question did have a significant amount of native code. We investigated the possibilities by which an application can become unstable. Our efforts to reproduce the problem were largely unsuccessful. We found some useful literature that stressed the importance of doing static analysis of native code, allowing/disallowing signal handling routines between native code and the underlying OS, the potential for collisions, etc. Since the native code was loaded by the application server and not by the application, there was very little we could do in terms

of analyzing and monitoring the suspect code. Such debugging experiences are not uncommon and they emphasize the importance of isolating applications and their native libraries from each other. That begs an understanding of how the J2EE platform provides for isolation between multiple concurrent applications. A J2EE server uses the classloader mechanism to provide isolation between these enterprise applications and it has its limitations, which we'll discuss later. A J2EE server virtualizes the underlying resources to an application similar to an operating system. It makes it possible to share JDBC connections between applications and some application servers allow the creation of connection pools per application. It provides life-cycle services to components and applications but doesn't have a way to stop threads safely. It doesn't have the means to control CPU or to renegotiate memory, storage, etc. As a consequence of these limitations and from the fear of contention between co-located applications (discussed earlier), enterprises rely on the protection offered by the operating system. They deploy a single application on a single application server instance even on highly scalable SMP servers. They add more J2EE server instances for running different applications, resulting in overheads that take more system resources. This approach requires a separate administrative server to manage the server instances and that adds to the overheads further. This is not optimal and there has been a pressing need for applications to safely co-exist within the same J2EE server.

What Is Wrong with Classloaders?

Classloaders make it possible to load multiple instances of the same class. It's possible to modify the search mechanism for class files within the JVM. Classloaders act like a namespace whereby the classes loaded by a particular classloader are tagged to provide a

unique identity. With classloaders, it's also possible to dynamically modify the bytecodes just prior to their loading by performing certain transformations. An application can be unloaded by discarding its classloader. However, a classloader can only be discarded if the reference count for all the classes loaded by this classloader becomes zero and the garbage collector deems them as unreachable. Classloader share basic classes; this can be exploited to change the shared data, thus making the other classloader and its applications vulnerable. The security model is debatable at best. Here's an example: if there exists a utility class that's used by an application and that application is distributed across two separate classloaders, replicating the utility class in both of these classloaders is obviously expensive. It's certainly possible to create a hierarchy of classloaders and having the parent classloader load the utility class, thus allowing the two classloaders to share code. Isolation provided by classloaders is weak because objects can leak and be captured. This approach therefore is incomplete and error-prone.

Isolates

Isolates are Java APIs that provide a uniform mechanism for managing Java applications that are isolated from each other, but can potentially share underlying implementation resources. Each application is given the illusion of running in its own isolated virtual machine and they each have their own system properties, classpath, static class state, etc. There are two primary motivations behind this. One is to prevent the faults within an application from propagating to other applications. The other is to ensure that an isolate can be terminated. The primary platform targeted has been J2SE, although nothing within the API inhibits adoption in any platform. There has been a strong case for adoption within



Murali Kaundinya is a chief architect within the Enterprise Web Services – Application US Sub-practice at Sun Microsystems, Inc. Murali is interested in service-driven development and service-oriented-architectures. He is currently involved in building a reference Architecture for SOA. He is a Java Ambassador and is also an active member of Sun's Customer Engineering Technical Council. Murali is a frequent speaker at major Java and SOA conferences.

murali.kaundinya@sun.com

the J2ME. The specification is currently under public review. As an API, it doesn't influence the implementation too much. Isolates provides a means of deploying new Java implementation features that enable and enhance scalability while providing an alternative to ad hoc control schemes. A conformant implementation of the API must guarantee at least isolation of Java state (i.e., logically disjoint heap spaces, per application mutable static fields, etc). Java applications that span across virtual machine instances can be wrapped with the Isolation API by adding only a few mechanisms for control. It's also possible to have implementations of Java isolation that provide high degrees of class, bytecode, and native code sharing within the same VM or between multiple VMs. The specific features of the chosen implementation will be discernable via a combination of vendor-specific and standard command-line arguments and properties.

Isolates' Architecture

The Architecture for Isolates is shown in Figure 1. An Isolate is a class and is part of the Isolate API. Isolates have their own security manager, application classpath, and disjoint heap. They share the statics of all of the classes with AWT. Isolates cannot share any objects between them. An Isolate has a halt () method that unlike the Thread.stop () method causes all the threads within the Isolate to stop and keep the state consistent. It's still recommended that this approach be pursued after attempts to shut down an Isolate have failed. An Isolate can communicate with another Isolate through a new mechanism called a Link, which is a class that is part of the Isolate API. A Link allows I/O objects such as files, and sockets to be passed between Isolates. RMI can just as well be used and in most cases may be preferable to Link. Thus each Isolate acts as a separate logical virtual machine. An Aggregate is a logical group of Isolates that share the runtime executable bytecodes and the class representation of all the classes within. An aggregate can contain one or many Isolates; however, it's not a class. Aggregates generally share state that is managed or owned by the operating system. State such as the "current working directory", the hostname, the user ID who launched the JVM, etc., are outside the JVM and are considered to be state at the aggregator level. Isolates can use all forms of external communication including RMI and RMI-over-IIOP.

Implementation Styles

As most Java APIs, the Isolate APIs don't dictate a whole lot on the implementation. Naturally, there are multiple ways to implement these APIs. One form of implementation would be to have an Isolate on its own OS process. This would be no different than existing applications except that they would be wrapped within an Isolate. The sharing with other Isolates would occur via interprocess communication mechanisms such as shared memory, message passing, and sockets. Class representations, bytecodes, compiled code, immutable statics, and other internal data structures can be shared in this fashion. The reference implementation of JSR-121 follows this style. Imagine having a single Isolate in Figure 1 – that would be a good example of this style of implementation. Another form of implementation would be to have all Isolates within one OS address space or a process. In this style, the Isolates still get their own versions of all the statics and global definitions including AWT thread and shutdown hooks. The Multi-tasking Virtual Machine and the Janos VM implement this style. Figure 1 is a good illustration of this style. A third form of implementation is to have Isolates scheduled to run on multiple JVMs. SAP follows this implementation style. Another style would be to have a LAN cluster of JVMs and have Isolates run on different hosts but all sharing a common administrative domain.

Isolates's API

It might be useful to get an appreciation for some of the design goals before jumping into the core of the Isolates API. The goals of this JSR have been to keep the APIs minimal and small. Such a goal can only be fulfilled if the semantics are precise and simple. The

APIs stress the mechanism but are not intrusive to dictate implementation styles and policy. As in all new Java APIs backward compatibility is an important requirement. This implies that there should be no changes to the code prior to JSR 121. The JSR Expert Group has also taken care to ensure that they allow multiple mapping strategies to different platforms. The Java docs for the Isolates API are shown in Figure 2. The JSR is still under development and therefore these might still undergo some changes. For the most part, the APIs are quite stable.

The API consists of the **Isolate** class, which can be thought of as a handle to an isolated computation. It has a few constructors, one of which takes the IsolateParameters class as an argument. The IsolateParameters class wraps the command-line parameters to indicate the style of implementation and other runtime properties. Isolate object instantiation corresponds to preparation of an isolated computation (an application). The Isolate class has methods for starting, suspending, resuming, and terminating the isolated computation at present, as well as methods for determining state, waiting for termination, and determining familiar relationships of computations. The other main class defined by this JSR is the **Link** abstract class, which can be thought of as a bidirectional pipe between two Isolates. The Link class has a method to create a link with a sender and a receiver and a method to close the link. It also has methods to send, receive, and check if the link is open, and utility methods to check if an Isolate is a sender or a receiver. Most of the classes including Isolate and Link implement a marker interface called Message. There are various classes that implement the Message interface such as CompositeMessage, DataMessage,

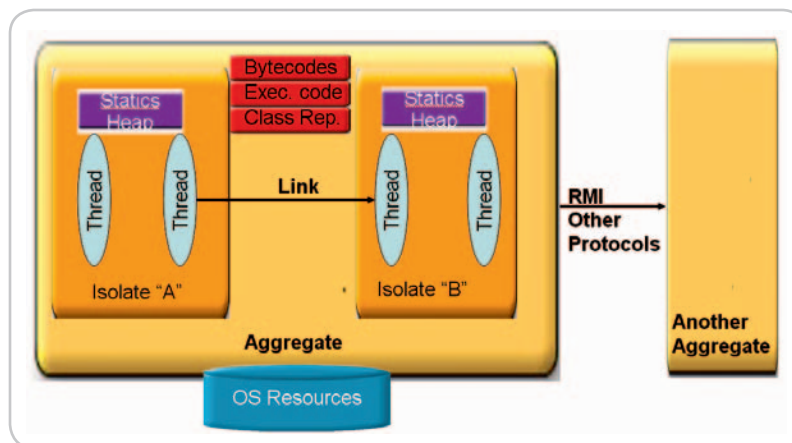


Figure 1 Aggregates vs Isolates vs Threads

and StatusMessage. ByteBufferMessage, ChannelMessage, FileMessage, and SocketMessage implement IMessage, which extends the Message interface. They all provide the abstractions as indicated by their individual prefixes.

Isolate operations that alter state make security checks that can throw security exceptions. The IsolateStartException class is a wrapper for the regular Exception. Isolate objects to access existing computations can be obtained through a static lookup method using identifiers that are unique within the platform's domain (e.g., a computer, a cluster, etc.).

Examples

Let's take a look at a few examples:

```
void runProgram(String classname, String[]
args) {
    try {
        Isolate i = new Isolate(classname,
args);
        i.start();
    }
    catch (SecurityException se) {...}
    catch (IsolateStartException ise) {...}
    catch (Exception exc) {...}
}
```

As described in the code snippet, creating an Isolate and running a computation within an Isolate is not very different from running an application today (without Isolates). If there is a class called MyApp, accessible in the default application class-path, passing the string "MyApp" to the runProgram method would run the "MyApp" within an Isolate. As discussed previously, each Isolate has its own application classpath and this can be modified just prior to creating and running an Isolate as shown below:

```
TransientPreferences tp = new
TransientPreferences();
tp.put( "java.properties/java.class.path",
class path");
try {
    Isolate myIsolate = new Isolate("MyApp",
null, tp, null);
    myIsolate.start();
}
catch (SecurityException se) {...}
catch (IsolateStartException ise) {...}
catch (Exception exc) {...}
}
```

The following code snippet shows a J2ME application wherein a parent and a child Isolate communicate with each other using a Link.

```
Class Runner {
    Link data;
    Isolate child;
    CompositeMessage getMessage() { return
data.receive(); }
    StatusMessage runStarlet(String mCls,
String[] mArgs,
                                String[] sec
/*,...*/) {
        IsolateParameters context = new
IsolateParameters(mCls, mArgs);
        context.setContext(
            "jsr121.exp.java.properties.java.
security.manager",
            sec);
        child = new Isolate(context);
        data = Link.newLink(child, Isolate.cur-
rentIsolate());
        StatusLink s = child.newStatusLink();
        child.start(new Link[] { data } );
        return s.receive();
    }
}
```

Summary

We have seen that Isolates (JSR 121) provide a mechanism to launch multiple applications and a natural consequence of this is that the Java environment is not dependent on the shell scripts of the operating system. In a way, they become platform neutral. This is also good from a security perspective as the launching code can be inspected and verified for its type safety and potential for buffer overflow. Whenever they become part of the platform, Isolates can avoid launching multiple JREs and this can improve the scalability of the platform. As each Isolate runs almost like a logical virtual machine, it's conceivable to start and spawn Isolates depending on the load of the application, the resources on the host machine, and more. JSR 121 has completed its Public Review and the Executive Committee for SE/EE has approved the JSR Review Ballot for JSR 121. This article does not imply product plans or any such commitments to the Java platform whatsoever.

Acknowledgments

I would like to thank Pete Soper for providing a great deal of insight on JSR 121 and for clarifying many of the questions on the specification. I also thank Ali Syed for reviewing the initial draft of this article and providing feedback. ☺

References

- *Application Isolation in the Java(tm) Virtual Machine*, Grzegorz Czajkowski, ACM OOPSLA'00, Minneapolis, MN, October 2000.
- *Automated and Portable Native Code Isolation*, Czajkowski, G., Daynes, L., and Wolczko, M., Sun Microsystems Laboratories Technical Report 01-96, April 2001.
- *Experience with Secure Multi-Processing in Java*, Dirk Balfanz, Li Gong, IEEE Proceedings of ICDSC'98, Amsterdam, The Netherlands, May 1998.
- *Building a Java virtual machine for server applications: The Jum on OS/390* Dillenberger et al IBM Systems Journal (Java Performance issue), Volume 39, No. 1, 2000.
- <http://www.bitser.net/isolate-interest/papers/bryce-05.04.pdf>
- <http://www.jcp.org/en/jsr/detail?id=121>


Packages

[javax.isolate](#)
[javax.isolate.io](#)
[javax.isolate.nio](#)
[javax.isolate.tbd](#)
[javax.isolate.util](#)

All Classes

[ByteBufferMessage](#)
[ChannelMessage](#)
[CompositeMessage](#)
[DataMessage](#)
[DataMessageInputStream](#)
[DataMessageOutputStream](#)
[FileMessage](#)
[IOIsolateParameters](#)
[IMessage](#)
[Isolate](#)
[IsolateParameters](#)
[IsolatePermission](#)
[IsolateStartException](#)
[Link](#)
[LinkClosedException](#)
[Message](#)
[MessageDispatcher](#)
[MessageHandler](#)
[MessageVisitor](#)
[ObjectMessage](#)
[SocketMessage](#)
[StatusMessage](#)

Figure 2 java docs



Does your
organization
have an
IT Super Hero?

Deploy *Intelli***VIEW**... and you can be the one!

IntelliVIEW is an easy to use interactive reporting solution that allows IT to create, deploy and distribute reports in a matter of minutes.

With IntelliVIEW you can:

- Connect to any database
- Generate ad-hoc reports
- Provide powerful analytics


- Integrate seamlessly into web applications
- Define user access levels
- Schedule and email reports automatically

These combined with its uncomplicated licensing and low TCO give you the power to be the IT Super Hero of your organization.

Be the IT Super Hero of your organization

call toll-free **1-866-99IVIEW**
or visit

<http://www.intelliview.com/jdj>

Product of
 **Synaptris**

3031 Tisch Way, Suite 1002, San Jose, CA 95128. USA

Coexisting in the Java Universe

An interview with Anthony Scotney, chief technical officer of WebRenderer's development company, JadeLiquid Software

Interview by
Jeremy Geelan

Anthony Scotney is a science graduate of the University of Tasmania with majors in computer science and information systems. After graduating he established JadeLiquid Software Pty Ltd. to develop software tools that would enhance the Java programming language. JadeLiquid Software has received several awards and grants in recognition of its success including Tasmanian Export Award Finalist 2003 and 2004 and awarded the 2004 Microsoft Emerging ICT Business Award. Anthony himself was awarded the prestigious 2004 Pearcy Award for innovative and pioneering achievement and contributions to research and development in information technology. He also was named 2005 Tasmanian Young Achiever of the year.

JadeLiquid Software Pty Ltd. is a leading provider of Java rendering components for enterprise applications and Web Services. WebRenderer can connect a client application to server business logic without changing server code or infrastructure. WebRenderer is the only standards-compliant Java Web content rendering component available commercially. JadeLiquid has a client base spanning the globe with deployments in North America, Europe, and Asia.



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

Q Thanks for agreeing to talk with JDJ. Let's cut right to the chase. Perhaps you would start by unpacking the concept of a "wrapper library" itself, and then follow that general definition by situating WebRenderer in the overall Java browser space.

The challenge for Java rich-client applications is to display the rich Web content in a meaningful way. Java doesn't have a standards-compliant Web content rendering engine built into the J2SE SDK. This poses a challenge for Java developers. Java rich-client application developers were traditionally forced to use the default

browser on the system external to their rich-client application to display rich Web content in a standards-compliant and reliable manner. WebRenderer addresses this limitation by providing an embeddable standards-compliant Web content rendering engine.

The concept behind WebRenderer was to build a Java browser component without reinventing the wheel. So we went about designing a Java browser component that would leverage off the standards-compliant and industry standard browsers already in existence. We started by writing code that connected to Internet Explorer and brought the browser through to Java. We then moved to bring Mozilla and Safari through and started supporting multiple platforms. The goal has always been to make the component as easy to use as possible and to ensure it is deployable just like any other Java component. The version 3 release of WebRenderer was really the culmination of years of intense development by the WebRenderer team with a specific

focus upon making WebRenderer a Java browser component that is standards-compliant, Swing-compatible, easy to use and deploy.

When WebRenderer was first released there was only one serious player in the Java browser space. We saw that the space was somewhat stagnant and decided that we would re-energize the market through a Java browser component with a unique and innovative architecture. We have been able to do that and WebRenderer is now the top-tier highest-quality Java browser component. While competitors have sat stagnant for years our WebRenderer revenues have more than doubled year-on-year since release. Through the competition provided by WebRenderer we have made Java browser components more accessible to the Java community by forcing top-tier prices down by as much as 80%. The client Java space is an interesting and exciting market space to be in as we go through the re-emergence of Java rich clients. The



Anthony Scotney



Powering eclipse to the next level...

NitroX™

PROFESSIONAL TOOLS FOR ECLIPSE

JSP | Struts | JSF

Professional development for JSP, Struts & JSF
Simultaneous visual/source Struts & JSF configuration
Code completion for all levels
Automatic validation & error checking for all levels
Debugging support for all levels
AppXRay, AppXaminer, AppXnavigator

download at: www.m7.com/power



Top Reviews Winner: InfoWorld (8.3), CRN ★★★★★, SDMagazine ★★★★★

Java community was focusing so hard on Web Services that we knew it was only a matter of time before there would be a client-Java re-emergence to drive the presentation layer of the distributed architecture.

Q *Why do you think Sun has never built a Web rendering engine into the Java SDK itself?*

Sun actually developed a Java browser engine (HotJava) but it was dropped a long time ago. They kept a skeletal HTML5EditorKit for the SDK that offers basic HTML 3.2 rendering. I assume that Sun found the project too resource-intensive and in a way probably felt like they were reinventing the wheel years after Microsoft and Netscape released browsers. With Web standards evolving the resources required to develop and maintain a standards-compliant rendering component would have been larger than that required to maintain the entire Swing and AWT toolkits.

Q *Presumably WebRenderer supports JavaScript and XHTML? How about advanced CSS (CSS2)? Maybe you should just list the standards v3.0 complies with.*

WebRenderer is the most standards-compliant Java browser component commercially available. The standards compliance comes from the underlying use of Mozilla technologies. The Mozilla team has followed the W3C specifications (and other standards) to a “tee” while building Mozilla. So anything that Mozilla supports WebRenderer inherits. I guess you could say WebRenderer has good parentage. Some of the standards WebRenderer supports are: HTML 4.01, CSS 1 & 2, JavaScript, XML, XSL, XSLT, XHTML, and SSL.

Q *How does WebRenderer differ specifically from other Java HTML renderers like, say, Javio or NetClue?*

The challenge of rendering Web-content comes by virtue of the number, complexity, and evolving

nature of Web standards. A basic Web-content rendering component must support base standards such as HTML, CSS, JavaScript, and SSL. Without support for these base-level standards Web content will either not render or render in a non-standards-compliant manner leading to an unexpected visual appearance. WebRenderer addresses the rendering challenge by leveraging off Internet Explorer, Mozilla, and Safari browser technologies to provide a predictable and standards-compliant rendering of real-world Web content.

Given that WebRenderer is a layer above Mozilla/Internet Explorer/Safari it supports all common Web standards and the rendering predictability and faithfulness is greater than its pure Java counterparts. It is an extremely time-intensive task to write a pure Java implementation of, say, Internet Explorer, which has had hundreds of people and who knows how many hours spent on its parsing and rendering. Expecting any small company to deliver standards-compliant rendering from its own implementation is an unrealistic expectation.

WebRenderer also has a very complete API that allows developers to easily extract basic right through to advanced functionality without being overly complex.

Q *What's support for Swing like?*

We have spent years perfecting WebRenderers Swing support. As of WebRenderer version 3 Swing compatibility is excellent.

Q *Is it suitable for embedded devices, or is the desktop your main market?*

Initially we spent some time on an R&D pilot of WebRenderer for embedded devices. While we found that we could port WebRenderer to embedded devices, we found that there was little market demand. So we focused on the desktop market. With

the release of WebRenderer Server Edition we are supporting headless server environments and directing some attention to server-based Enterprise Java.

Q *What kind of organizations are using WebRenderer right now?*

WebRenderer is in use in many large and small organizations throughout the world. Given WebRenderer is a component the verticals are spread across many industries such as aerospace, semiconductor, manufacturing, technology, government, education, and finance.

Some of the organizations deploying WebRenderer include Cisco, Northrop Grumman, Lockheed Martin, the U.S. Department of Defense, the European Space Agency, the European Patent Office, Groxis, and Encyclopedia Britannica.

Q *Can individual developers download a trial version, or does their organization have to buy WebRenderer outright for them to be able to try it out?*

WebRenderer is available for a free 30-day trial.

Q *What kind of support do you offer to users?*

Our first line of support is our well-documented Developers Guide, API, and technical examples.

We provide a complete pre- and post-sales support service that involves customization, integration, and deployment assistance and technical issue resolution.

Q *Last, we all know that the Chinese call tea “liquid jade” but what's the connection between the poet Lu Yu, who first came up with the notion about the year 780 and your endeavor well over a thousand years later? ;-)*

The JadeLiquid name is fitting for the architecture choice of our components, not pure-Java, but it happily coexists. ☺

“The challenge of rendering web content comes by virtue of the number, complexity, and evolving nature of Web standards”

Style Report 7

Light Weight, Integration Ready Enterprise Reporting & Analysis



open standards
 open architecture
 Linux
 Windows
 Java
 J2EE
 Tomcat
 SOAP
 LDAP
 DHTML

Traditional BI Challenges:

- ▶▶ Monolithic, resource intensive
- ▶▶ Proprietary software stack
- ▶▶ Requires ETL/Data warehouse

Style Report Solutions:

- ▶▶ Light weight, integration ready
- ▶▶ Open software stack, J2EE drop-in
- ▶▶ Real time transactional DB and OLAP



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

What's New in Eclipse 3.1

Polishing the apple

by Ed Burnette

Since Eclipse's first release in 2001, it has become a popular environment for Java development. In the period between March 10 and May 11, 2005, users downloaded over 17,000 copies of one of the production SDK releases and over 3,500 copies of one of the stable (milestone) SDK builds on average *every day*. A vibrant eco-system of developers, plug-in providers, authors, and bloggers has grown up around it. Eclipse has also gained the backing of the key Java vendors including BEA, Borland, IBM, SAP, and Sybase. Developers like Eclipse because it provides a great platform for building Java applications, and companies like it because it unifies their software tools under one open source umbrella.

In late June of this year, the latest release of the Eclipse Platform, version 3.1, will be available for download from eclipse.org. In this article, I'll highlight some of the more interesting new features it contains. I'll also discuss some of the other Eclipse projects that are releasing new versions at about the same time.



Ed Burnette is the author of the Eclipse/DE Pocket Guide (to be published later this year by O'Reilly), co-author of Eclipse in Action, and editor of the articles section at eclipse.org. He writes about Eclipse and the Rich Client Platform at his Web site, www.eclipsepowered.org. Ed has programmed everything from multi-user servers to compilers to commercial video games since earning a BS in computer science from North Carolina State University. He is a principal systems developer at SAS, and lives near Research Triangle Park, North Carolina, with his wife, two kids, and a whole bunch of cats.

ed.burnette@sas.com

A New Hope for Developers

One of the major new features of Eclipse 3.1 is full support for the new language constructions in J2SE 5.0 (also called J2SE 1.5 in the old numbering scheme). Generics, annotations, enums, auto boxing, enhanced for loop, etc., – it's all in there, both in the underlying compiler and the user interface and code assistance that Eclipse is known for.

While Eclipse didn't invent the idea of refactoring, it provides one of the most complete implementations. Eclipse 3.1 comes with a number of new and enhanced refactorings, code assistance, and "quick fixes", many in conjunction with its J2SE5 support. For example, you can put your cursor on a conventional *for* loop that iterates over an array (see Figure 1), press Ctrl+1, and Eclipse will offer to convert it to one of the new style *for* loops (see Figure 2).

At the heart of Eclipse's Java support is a fully compliant incremental Java compiler, written in Java and supporting Java language levels 1.3, 1.4, and now 5.0. Having its own compiler brings Eclipse some benefits including fast compilation, smoother debugging and refactoring, and a lot of diagnostic warnings. The compiler has found several uses outside of Eclipse. It's bundled with many

popular Linux distributions and commercial applications, and recent versions of Apache Tomcat use it to compile JSPs. It forms the basis of the AspectJ compiler. And I wouldn't be surprised to see the Eclipse compiler used in the recently announced Apache Harmony project as well.

Other usability enhancements make 3.1 more productive. For example, the new release contains a more integrated help system that changes to show help for what you're doing at all times. One of the largest improvements is in the area of Preferences. Addressing a key user request, the Preference dialog now offers the ability to filter by keywords, for example, you can easily find all options having to do with "tabs" by typing that keyword into the filter box. In addition, Web-like navigation has been added to link to related preferences and go forward and backward in the history.

To make preferences easier to find, in Eclipse 3.1 the Preferences dialog can be opened directly from many editors and views through the context menu. For example, if you right-click in the Java editor and select Preferences..., the dialog will appear. Only the options related to Java editing, including those for the text editor that the Java editor inherits, are shown.

Eclipse 3.1 improves its Ant support by including the latest version of Ant, and an Ant script debugger (see Figure 3), plus many editor enhancements. Another welcome addition: the ability to import a project from an Ant build file, and to export and generate a build file from an existing Eclipse project – you can synchronize your CLASSPATH and build.xml with a few clicks. The generated build.xml is simple and clean, with a provision for a build-user.xml that you can override and still keep the benefits of build.xml generation. This is another example of the community in action: the import/export feature is based on the contribution of Richard Höfter, author of the `eclipse2ant` plugin.

One thing to note is that all these new features don't come with a performance penalty. Eclipse 3.1 is a lot faster and uses far less memory for common operations than version 3.0. Don't believe me? Check out the performance tests results on the download page for any recent build. These improvements are not just for Windows; Mac and Linux users will notice even if even more due to the special attention paid to those platforms. The graphs

WebRenderer™



Standards compliant embeddable web browser component

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

Why WebRenderer?

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

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

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

JadeLiquid™
Software

Copyright JadeLiquid Software 2004. JadeLiquid and WebRenderer are trademarks of JadeLiquid Software in the United States and other countries. Sun, Sun Microsystems, the Sun Logo and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other trademarks and product names are property of their respective owners.

don't tell the whole story, however. In normal day-to-day work I've found Eclipse 3.1 to be much snappier than any previous version.

With developers working with ever larger and more complex projects, their IDE needs to keep up. In order to experience and study problems with large workspaces, the Platform team created one consisting of 135 separate projects and 70,000 classes and other resources. Then, using various profiling tools they identified and corrected many bottlenecks, mainly in the area of memory usage and I/O. As a result, Eclipse 3.1 can handle bigger problems in less time than before. Launching the test workspace used to take close to two minutes, but in Eclipse 3.1 it now takes under 10 seconds.

Return of the Java Client

Java started out on the desktop, and now after a brief vacation on the server side, it's returning to the desktop with a vengeance. The Rich Client Platform (RCP) is helping to spark this renaissance. RCP is a subset of Eclipse that provides a framework for application development. It includes a widget toolkit (SWT), the plug-in loaders, the help system, and other components that you can use in your own programs.

By taking advantage of this free "client middleware," you can focus on your core competencies and reduce your time-to-market. Eclipse's corporate-friendly license (EPL) allows you to reuse the code in your own programs,

whether or not they are open source. You can modify and redistribute the code, as long as you return any improvements to the community.

The biggest change for RCP in Eclipse 3.1 is a set of wizards and editors for creating, building, branding, and deploying RCP applications. To create an RCP application just create a plug-in project, click the checkbox that says "Create an RCP application", select a template, and then click Finish. With a few more clicks you can export the project to create a deployable application. No more trying to figure out plug-in dependencies, tweaking configuration files, and copying plug-ins by hand. All that's handled for you in the new release.

Branded applications are supported through the new Product Configuration editor. You can change the window titles, icons, splash screens, and other branded elements of your program quickly and easily. And with the RCP Delta Pack you can create deployable packages for all supported platforms at the same time (see Figure 4).

RCP applications can take advantage of dynamic plug-ins, that is, plug-ins that come and go at runtime. This provides flexibility to the RCP application delivery model. A large application can be deployed progressively as plug-ins are loaded or on demand when extra functionality is needed. This technology was originally designed for mobile phone provisioning as part of the OSGi Service Platform, and later implemented in Eclipse by the Equinox project team. Eclipse is an active participant in OSGi, and Eclipse 3.1 includes several features slated for version 4 of the OSGi standard.

In one proof-of-concept example shown at EclipseCon, the developers demonstrated a calculator program that started out with only a plus and minus button. Using Eclipse's update manager and dynamic plug-ins, the calculator then downloaded a new plug-in that added a multiply button. All this is done in the running JVM process without a restart.

One of the most frequently asked questions about RCP-based applications is if you can deploy them with Java Web Start. The answer in Eclipse 3.1 is yes. New feature export wizards make this easy; they'll even sign the JARs for you and create a template .jnlp file. In support of Java Web Start, most Eclipse plug-ins have been converted to regular old Java .jar files. Information about extension points, plug-in dependencies, and so on go in manifest files inside the JARs.

In Eclipse 3.1, client developers can take advantage of a slew of UI improvements to make their applications even more functional and better looking than before. For example, SWT includes two new widgets: a Spinner widget for numeric data entry and a Link widget that allows hyperlinks to be included in text labels. A number of other widgets were enhanced.

The Tree widget now supports columns, deprecating the older TableTree widget. This allows a native implementation and helps resolve some of the more subtle problems with the TableTree, including the inability to add an image in the first column. Also the Table widget got a much requested feature: the ability to drag and drop columns to reorder them within the table. Virtual tables with deferred loading are also supported.

```
public class Example {
    public static void dump(String[] names) {
        for (int i = 0; i < names.length; i++) {
            String name = names[i];
            System.out.println(name);
        }
    }
}
```

Figure 1 Press Ctrl+1 to convert a normal array iteration loop.

```
public class Example {
    public static void dump(String[] names) {
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

Figure 2 After conversion, the code uses the enhanced |J2SE5 for loop.

Figure 3 The Ant debugger lets you single step and examine variables in Ant scripts as you would a Java program.

Smart Development Environment

15th Annual

Jolt Product Excellence Award
in the Design Tools category

Winner

SDE won the 15th Software Development Magazine Jolt Product Excellence Award 2005 in Design Tools category over IBM Rational Software Architect and Borland Together Designer.



VP Suite is available for a free 30-day trial at www.visual-paradigm.com

Visual Paradigm products



2004 Quadruple award winning product



Visual Paradigm helps
ACCELERATE the entire
MODEL-CODE-DEPLOY
process in a **DISCIPLINED** and
COLLABORATIVE way to **EXCEED**
customers' expectations.

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

 Visual Paradigm

The Browser widget continues to get attention as well. This widget wraps the native HTML browser on the current platform (for example, IE on Windows and Safari on the Mac). There have been numerous minor enhancements to the browser including many to its event mechanism. Perhaps the most exciting feature is the ability to execute an arbitrary string of JavaScript within the browser's currently loaded page.

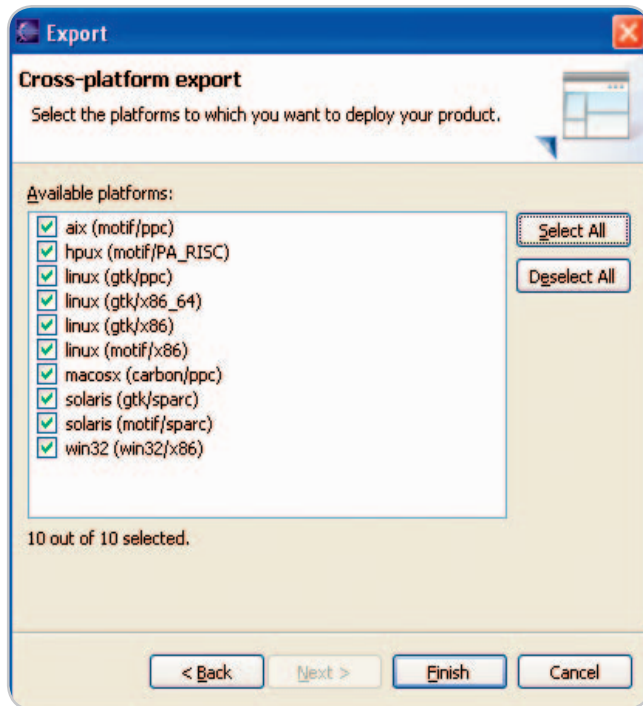


Figure 4 Create branded, deployable cross-platform products with a few clicks using the new export wizards.

Another area that was improved is SWT graphics. Eclipse 3.1 supports alpha-blending, anti-aliasing, paths for geometric shapes and lines, and transformations (see Figure 5). On Windows, using these GC new features takes advantage of the Microsoft GDI+ library (which is included with Windows XP and 2003 but available as a separate download on older systems). On GTK and Motif, the new graphics are implemented with the Cairo graphics library.

The Community Strikes Back

Community involvement is crucial to the success of Eclipse. One of the things you're expected to do as a good Eclipse community citizen is report your ideas for enhancements along with any bugs you find. Since the source code is available, some take the next step and send code patches as well. Over 7,000 enhancement requests and bug reports have been addressed in release 3.1.

The Eclipse community continues to grow through the addition of new projects. As of this writing, over a dozen new project proposals are pending or have been recently approved. Many of these are "Technology" projects, which are often created by groups of users that band together to fulfill a need. For example, the Mylar project was created at the University of British Columbia to address the problem with information overload by filtering out uninteresting classes and other artifacts while you're programming.

Following on the heels of the 3.1 release of the Eclipse Platform, a number of other Eclipse projects is expected to be released. One of the biggest, the Web Tools Platform project, or WTP for short, is scheduled to release a new version in late July. WTP was initially based on contributions from IBM and ObjectWeb, but many companies and individuals in the community are working on it now, including recent joiner BEA.

The Web Tools Platform currently has two subprojects: Web Standard Tools (WST) and J2EE Standard Tools (JST). WST provides a common infrastructure for Web applications development and provides editors, validators, and document generators for a wide range of Web languages (HTML/XHTML, CSS, JavaScript, Web services, SQL, XML, XSD, WSDL, etc.). You can also publish and deploy, run and debug, start and stop Web applications on target servers (see Figure 6). WST also includes a TCP/IP Monitor server for debugging HTTP traffic (including SOAP Web services), and a Web services explorer that is very handy for testing. Currently it also has support for relational databases management and queries, though that may be moving to the new Data Tools project soon.

JST extends WST for J2EE applications and servers. Included is a range of tools simplifying development with J2EE APIs including JSP, JCA, JDBC, JTA, JMS, JMX, JNDI, and Web services. It builds on WST to support J2EE servlet engines and EJB containers, including Apache Tomcat, Apache Geronimo, and ObjectWeb Jonas. Server vendors are encouraged to develop adapters for their servers.



Figure 5 SWT now supports alpha blending and anti-aliased text (see inset).

Think ~~Java development is complex & overengineered?~~

~~Think~~ **again.**



~~I want to be known for consistently delivering applications and providing business value.~~

~~That is why I use ThinkCAP.~~

ThinkCAP™

~~ThinkCAP 6.0 simplifies and accelerates the development and maintenance of J2EE-based web applications by 50%.~~

~~ThinkCAP's Visual Workbench and integrated MVC framework bring high productivity to corporate application developers (those with PowerBuilder, RPG, or Oracle-like skills) while letting J2EE programmers build reusable business logic & components using their tools of choice. The result is productivity for the whole development team.~~

~~Because of features such as Smart Data Binding, Visual Page Flow, and seamless Hibernate/Castor support, your team focuses on value added functionality — not low-level "plumbing." ThinkCAP integrates over 20 open source projects providing a practical mix of high productivity, robust functionality, and standards support.~~

Download a free trial version or view a demo at www.clearnova.com.

CLEARNOVA
Deliver Web Applications at the Speed of Business

Highly Productive Visual Workbench

Actions-based MVC Framework

Page Flow Designer & Generator

Easy to Learn Event Model

Advanced Data Aware Components

Forms, DataViews, Queries, Navigations

Workflows, Graphs, Treeviews, Grids, Tabs

Multirow Updatable DataViews

Extensible Browser & Server-side Validation

Point & Click Smart Data Binding™

Service Flow Designer for Web Services/SOA

Content Management & Workflow Engine

Platform Independence: Any Java Server/OS/IDE/Database

Integrated Role-Based Security

Seamless Integration with over 20 Open Source Libraries

Team Development with CVS Support

“ No more trying to figure out plug-in dependencies, tweaking configuration files, and copying plug-ins by hand; all that is handled for you in the new release”

Another widely anticipated project is the Business Intelligence and Reporting Tools (BIRT) project. BIRT 1.1 is targeted for July, and it will be based on Eclipse 3.1. Currently BIRT includes three components:

- A Report Designer for developing and designing XML report templates
- A Report Engine for generating reports based on the XML template (you can use it standalone or embedded in other applications)
- A Chart Engine for creating charts within BIRT reports or as a standalone API to draw charts in your Swing or SWT applications.

Future plans for BIRT include a Web-based Report Designer.

The Eclipse Test and Performance Tools Platform Project (TPTP), formerly known as Hyades, will launch the 4.0 release in July as well. TPTP delivers components in four areas:

- A platform for building testing tools, with common UI components and standard data models
- Monitoring tools for things like analyzing a Web server
- Test tools, including support for JUnit
- Tracing and profiling tools

TPTP 4.0 delivers better integration with JUnit, new hooks to make it easier to link test cases back with requirements and defects, and usability improvements.

Visual Editor Project (VE) The Visual Editor Project will be releasing version 1.1 approximately two weeks after Eclipse 3.1. Highlights include:

- Support for new SWT controls

- Better support for Swing tables
- Copy/paste support
- Support for editing Eclipse views directly (especially useful for RCP programs)
- Better code generation and reverse parsing (produces code more like what you would write by hand)

The AspectJ Technology Project will release AspectJ 5.0 soon after Eclipse 3.1 is shipped. The new version includes full support for J2SE5 features, integration of AspectWerkz-style code, better deployment (especially for container-based environments), faster performance, and more comprehensive IDE support. For example, generics are integrated with AOP language features such as join points, pointcuts, advice, and inter-type declarations. Annotations bring AOP to pure Java source files, so you can continue to use your favorite Java compiler and then weave in the aspects in another build step or when classes are loaded. Deployment in J2EE containers is easier and compiling and weaving runs faster and generates better code than before. The class-loading and runtime aspect weaving that made AspectWerkz so convenient should also be supported.

For a gentle introduction to AOP, you may want to check out the Concern Manipulation Environment project (CME) project. It offers powerful code navigation to help you identify cross-cutting aspects in your existing Java code.

Finale

In four short years since Eclipse exploded onto the scene, it has come to dominate the Java IDE landscape. User groups have sprouted up around the world, and hundreds of books and articles have been written about it (two dozen in Japanese alone!). Eclipse 3.1 is the culmination of a year's worth of development effort on features such as J2SE5 support, performance improvements, and rich clients. If that weren't enough, it will be the base of the next wave of software releases from the Eclipse Foundation and its partners. Whether you're a programmer trying to build the next Killer App or an entrepreneur building a business model on open source, this is an exciting time to be involved with Eclipse.

Acknowledgments

I wish to thank the many readers of www.eclipse-powered.org who contributed to this article, including Chris Gross, Philippe Ombrédanne, Ng Chin Kiong, Sam Mesh, Bob Foster, David Orme, mgallego, Imandel, and nobodaddy. And a special thanks to Xavier Méhaut, who maintains the Eclipse wiki site, <http://eclipse-wiki.info>, where we worked on the draft. ☺

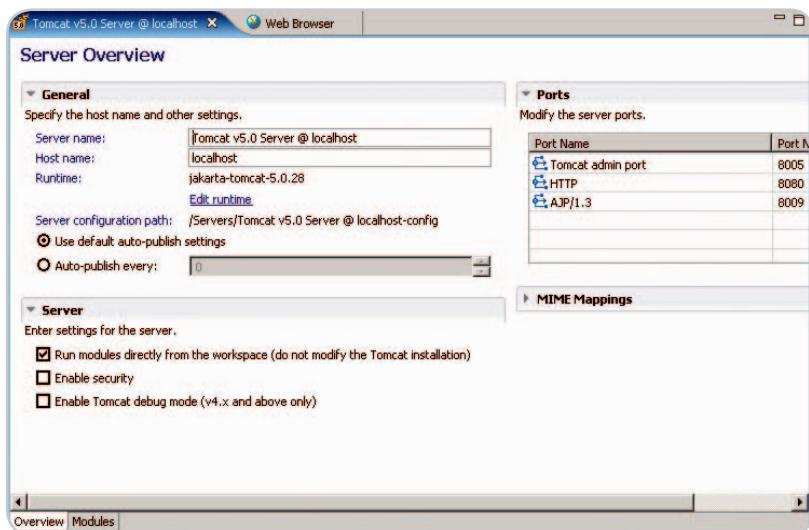


Figure 6 WST is expected to ship with support for several open source servers like Tomcat

Javavavoom!

Get a high-performance, transactional storage engine that's 100% Java.

Berkeley DB Java Edition 2.0

Download at www.sleepycat.com/bdbje

Now there's a high-performance storage engine 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. And because it supports J2EE standards such as JCA, JMX and JTA, you can be sure you have the widest range of options.

Experience the outstanding performance of Berkeley DB JE for yourself.

Download Berkeley DB Java Edition today at www.sleepycat.com/bdbje, and view the presentation "*Design and Implementation of a Transaction Data Manager*."





Calvin Austin

Core and Internals Editor

JavaOne from the Inside Out

This year will be the first time in 10 JavaOnes that I haven't been a Sun employee. As I am now fairly local to the show I should be able to attend again this year. I've met many developers from around the world who make the annual trip to San Francisco. Many still see it as the Java event to network at, even though attendance is off the highs of the dot.com days.

Now you may think that being a Sun employee automatically gets you a free pass to the JavaOne show. Well it doesn't. In the early days there were a small number of passes for JavaSoft engineers, but the only way we could attend was to prepare a session or an evening birds of a feather discussion. At the peak Sun engineers were even banned from buying a JavaOne pass to free up places for Sun customers and Java developers. So to accommodate the Java developers inside the company Sun runs a mini-JavaOne conference originally called Java Premier that included select talks from the conference. At its peak JavaOne was almost too big. It was impossible to get a hotel room months before the event and I remember Pat Sueltz, the VP of Java, asking in her keynote if anyone had a free room since she didn't have a place to stay that night. I don't think she was joking. I only managed to find a place with two other colleagues in a flea-bitten motel south of Market.

The JavaOne stage has seen a Who's Who of the Java world and beyond. One of the most memorable sessions for me was a talk by the late Douglas Adams whose cult classic "The Hitchhikers Guide to the Galaxy" is currently showing in theaters in the U.S. He

gave a great perspective on things and just not Java. I don't know who they've lined up as the special guest this year, sometimes availability isn't confirmed until very close to the event. I'll be checking MaryMaryQuiteContrary's blog for details of the keynotes and the after-show parties.

For the main keynotes, I would expect the usual presentations we've seen for the past five years, lots of statistics about Java including the number of downloads and an emphasis on phones and J2ME from Sun. The first day's keynote rarely starts on time so unless you have your wireless laptop (hint) you may be sitting for quite a while. The keynotes are often available by live webcast so check back with the JavaOne site nearer the time if you can't make the show.

One final thought on keynotes, I wouldn't be too surprised to see Microsoft on stage or with their own keynote session since they're apparently a sponsor this year. The last year I remember Microsoft having any real presence was at the first JavaOne

when they organized a free party at a local restaurant.

The rest of the day is made up of hour-long sessions. The limit on rooms makes session selection difficult, often a speaker will get at most one talk accepted and the temptation is to try and target the session at a middle ground, especially if the planners expect a large audience. This is one of the reasons the advanced sessions can never be that advanced. For Sun employees writing the presentations is more drawn out since there are trademark and other reviews that have to take place while meeting your own work deadlines.

My tip, if the presenter didn't cover the material you wanted to see I would recommend asking questions at the end (after they've cleared the stage to make way for the next presentation). More often than not they'll know the answer but couldn't cover it in the given presentation time. I was always happy to sit down over coffee if it came to that.

The evening brings the BOFs and after-dark entertainment. We probably won't see parties like the Iona Spinal Tap concert again but many of the smaller vendor events can be just as much fun and are useful for networking. Make sure you ask around the pavilion floor for events. Most will occur in the first two evenings so make sure you visit the pavilion early.

In closing if you don't get to attend the show look for slides and webcasts afterwards. There are many dedicated folks at Sun who work tirelessly to make JavaOne the best developer conference they can so I hope you enjoy it. ☺



“One of the most memorable sessions for me was a talk by the late Douglas Adams... he gave a great perspective on things, not just Java”

A section editor of JDJ since June 2004, Calvin Austin is an engineer at SpikeSource.com. He previously led the J2SE 5.0 release at Sun Microsystems and also led Sun's Java on Linux port.

calvin.austin@sys-con.com

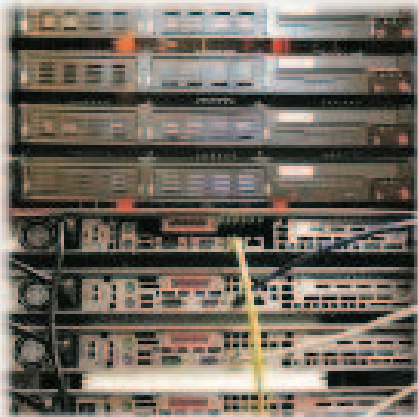
Complex J2EE Hosting made easy.

WebAppCabaretsm

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

\$49
monthly

Private JVM
Complete J2EE Hosting solution
for medium size web site.



\$279
monthly

Dual Xeon
2GB RAM
2 x 73GB
SCSI
2000GB
monthly
transfer.

J2EE and
Cluster
ready.



Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with Java J2EE.

At WebAppCabaret our standards based process and tools make deploying Java J2EE applications as easy as a point-and-click.

We call it **Point-and-Deploy Hosting**.

Our advanced NGASI Web Hosting management Control was designed for the hosting and management of complex web sites and applications thus cutting down on maintenance time.

Backed by an experienced staff as well as a **Tier 1 Data center** and network. Our network is certified with frequent security audits by reputable security firms.

All hosting plans come with **advanced tools** to manage your application server and Apache web server. Not to mention the other features, such as virus-protected email, bug tracking, and many more portal components.

Complete power and control at the tip of your fingers. We take care of the system and hosting infrastructure so you can concentrate on development and deployment of your application. That is **real ROI**.

Log on now at <http://www.webappcabaret.com/jdj.jsp> or call today at 1.866.256.7973



WebAppCabaretsm

Point-and-Deploy J2EE Hosting

Prices, plans, and terms subject to change without notice. Please log on to our website for the latest price and terms. Copyright © 1999-2005 WebAppShowcase • All rights reserved • Various trademarks held by their respective owners.

Kicking the Tires on Java 5.0

Building an aspect-oriented framework based on annotations

by Peter Braswell

I'm really jazzed about Java 5.0! We've been treated over the years to incremental improvements in JVM performance. JDK 1.2 brought us the collections framework as well as Swing, the thread context class loader, and improvements in RMI. JDK 1.3 and 1.4 continued in the same vein with logical improvements to libraries, JVM enhancements, and performance upgrades. Although this article doesn't intend to take trip down memory lane, it's important to understand that Java 5.0 brings a truly remarkable and rich set of new tools to our programming landscape as compared to other JDK releases.

This article will survey some of Java 5.0's new features and put them into practice through example. We'll build up a lightweight aspect-oriented system based on annotations to showcase what's new in 5.0. Some of these features you may be familiar with, some you may not. I've attempted to mix the obvious with some of the obscure. We'll examine some of the new hooks that the JVM has exposed for class loading, which makes the once dreadful work of bytecode manipulation during class loading much easier. In the "obvious" column, we'll look at generics and how they enable us to write more robust and sane programs, especially when dealing with collections. Perhaps the most notable aspect of Java 5.0 that we'll examine is the annotation framework. Annotations allow developers to inject metadata into their applications. We'll use this feature to demark classes we want manipulated at load time. To put this all in context, we'll create a lightweight framework that will manipulate classes as they are being loaded to enable logging, security, BAM (business activity monitoring), or any number of other scenarios that have yet to be dreamed up (The source code for this article can be downloaded from <http://jdi.sys-con.com>.)

Annotations

Annotations are nothing new. In concept we've been injecting forms of metadata into our programs for years.

If you've ever used XDoclet or EJBGEN to annotate a class in preparation for EJB deployment descriptor generation, you've used a form of annotation. Although these annotation methods are primarily manipulated during compilation, frameworks do exist that allow runtime access to annotations. Those that come to mind are the Jakarta Commons Attributes project and the metadata infrastructure in the Spring framework. One important thing to note from the Spring documentation regarding the use of Java 1.5 annotations versus metadata available in the Spring framework is the following:

"JSR-175 metadata is static. It is associated with a class at compile time, and cannot be changed in a deployed environment. There is a need for hierarchical metadata, providing the ability to override certain attribute values in deployment – for example, in an XML file."

For our purposes, however, the annotations suggested by JSR-175 and implemented in Java 5.0 will be sufficient.

Anatomy of an Annotation

Generally, annotations are thought of only as artifacts useful at compile time by tool vendors to do such things as generating deployment descriptors. This is what utilities such as XDoclet and EJBGEN do. The annotations are examined at compile time, used to generate output (in the case of EJB this maybe a deployment descriptor) and in a sense are then discarded thereafter as an artifact. Annotations in Java 1.5 can behave in a similar way, but they can also be retained past the compilation stage and accessed at runtime. The developer has three retention policies to choose from. They are:

- **RetentionPolicy.CLASS:** Annotations are to be recorded in the class file by the compiler but need not be retained by the VM at runtime

“Annotations are really nothing new. In concept we've been injecting forms of metadata into our programs for years”

Peter Braswell possesses over 15 years of software engineering expertise. He has led the implementation of leading-edge, mission-critical systems and applications in a variety of industries including high technology, finance, retail, and telecommunications. His solution and technology experience encompasses large object-oriented systems, distributed object infrastructure, and enterprise system integration. Peter holds a BS in computer science with a minor in philosophy from Old Dominion University.

pbraswell@alterthought.com

- **RetentionPolicy.RUNTIME:** Annotations are to be recorded in the class file by the compiler and retained by the VM at runtime, so they can be read reflectively.
- **RetentionPolicy.SOURCE:** Annotations are to be discarded by the compiler.

To declare a new annotation type, the developer must specify a retention policy for his or her annotation, what the element type is and what attributes the annotation possesses. The developer can associate an annotation with a particular element. Valid element types are:

- **ElementType.Constructor:** Associates an annotation with a constructor.
- **ElementType.Field:** Associates an annotation with a field.
- **ElementType.LocalVariable:** Associates an annotation with a local variable.
- **ElementType.Method:** Associates an annotation with a method.
- **ElementType.Package:** Associates an annotation with a package.
- **ElementType.Parameter:** Associates an annotation with a parameter.
- **ElementType.Type:** Associates an annotation with a type.

Let's look at a typical annotation declaration.

```
package annotations;

@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD})

public @interface BAMAnnotation
{
    String insertionPoint();
    String processBean();
}
```

What's notable here is that the annotation declaration is strikingly similar to an interface declaration and that the metadata for this annotation is defined in terms of an annotation! This particular annotation is used on a method and its values are accessible at runtime as dictated by the retention policy. The annotations require two attributes to be defined, "insertionPoint" and "processBean. To use this annotation in a class is pretty straightforward:

```
import annotations.*;

public class BizComponent
{
    @BAMAnnotation(processBean="nullInjector",
        insertionPoint="pre")
    public void execute()
    {
        System.out.println("Executing"+
            "some biz functionality");
    }
}
```

Here we've associated our annotation with the "execute()" method of our "BizComponent" class. When we examine this class at runtime the values of "processBean" and "insertionPoint" will be "nullInjector" and "pre" respectively.

For those of you who have figured it out already or perhaps

those of you who are wondering, what we are developing is a scheme by which we declare our aspect point-cuts via annotations. From a programming perspective we're dealing with an ordinary class (BizComponent) and we're using an annotation to tag it for bytecode engineering during the class-loading phase (more on this later). The nice thing about this approach is that the annotation (metadata) lives close to the source code eliminating the need to alter multiple source files. Compare this to "conventional" aspect-oriented programming where you have to declare this point-cut in a separate file and precompile or wire things up in an XML file like you would using Spring.

Getting Annotation Information at Runtime

From a programming perspective, getting information out of an annotation is pretty straightforward. Continuing with the example that we've developed to this point, imagine that we have a candidate class that we want to examine for annotations. We know that only our methods are annotated, so given a class, we first extract all the "Method" objects and iterate over them searching for our annotations, like so:

```
for( Method m : methods )
{
    ...
    Annotation [] annotations =
        m.getDeclaredAnnotations();
    for( Annotation a : annotations )
    {
        System.out.println("annotation type: "
            +a.annotationType());
        if(AMAnnotation.class.getName().
```

We've got problems with your name on them.

At Google, we process the world's information and make it accessible to the world's population. As you might imagine, this task poses considerable challenges. Maybe you can help.

We're looking for experienced software engineers with superb design and implementation skills and expertise in the following areas:

- 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 brain-bursting projects with your name on them in Mountain View, Santa Monica, New York, Bangalore, Hyderabad, Zurich and Tokyo.

Ready for the challenge of a lifetime?
Visit us at <http://www.google.com/jdj> for information. EOE



“Generics also support a tighter contract between caller and service, which is generally, always a good thing”

```

    equals(a.annotationType().getName())
    {
        // Byte-code engineering here...
    }
}

```

Note the use of the new for loop iterator. Gone are the days where you have to iterate over unsafe, un-typed collections. Happy Days indeed!

Once we have the Annotation object, getting properties is just as simple as this snippet of code demonstrates:

```

if( getAnnotationProperty("insertionPoint", a.toString()).
equals("pre"))
{
    ...
}

```

This code block simply pulls a property value out of the annotation and checks to see if it literally equals the string “pre.”

Great, What Happens Next? – Class Loading in 5.0

So now that we have our classes all annotated up, what do we do? Recall the intent of our Java 5.0 featured framework. We want to be able to annotate our classes (mark them) so that at runtime they are loaded in, examined, and altered (if they have the correct annotations) in such a way that a piece of code will run either before (“pre”) or after (“post”) our business method gets executed.

The class-loading mechanism in Java 5.0 has been extended and has provided us with a really slick feature: agents! Very simply, an agent class is something that you pass along to the VM via a command-line argument that lets you do things PRIOR to your main being called. Among other things, you can install a class-file transformer that’s invoked after the class has physically loaded into the VM, but before the class is handed back to the application. What this gives you is the ability to manipulate the class physically as it’s being loaded by the JVM. Before Java 5.0, this involved somewhat tricky, awkward custom class-loader wizardry. No more!

Here’s the full implementation for our Agent class:

```

import java.lang.instrument.*;
public class Agent
{
    public static void premain (String fqncEngineeringClass,
                               Instrumentation instrumentation)
    {
        try
        {
            // Install the new bytecode engineering loader
            ClassFileTransformer xformer = (ClassFileTransformer)Thread.cur-

```

```

rentThread().getContextClassLoader().getClass().forName(fqncEngineeringClass).newInstance();
            instrumentation.addTransformer(xformer);
        }
        catch(Exception ex)
        { System.err.println("Could not instantiate Bytecode Engineering Class: "+ex.getMessage()); }
    }
} // class Agent

```

There’s nothing special about an Agent class, other than it must contain the one static “premain(String, Instrumentation)” method. Besides that, it doesn’t have to implement any particular interface or extend any particular class. So in our case our agent code will get called in advance of our static main being executed. This requires some special arguments to the JVM that we’ll cover in a moment. When our premain gets executed, we’ll have two parameters passed in. One is the fully qualified name of our bytecode engineering class and the other is the hook that we’ll need to install it.

Notice that the premain method dynamically (via reflection) tries to instantiate the transformer class and then turns around and installs it as a class-file transformer. The class instantiated here is the class that implements the interface “java.lang.instrumentation.ClassFileTransformer.” In our case, this is the class that will do all the heavy lifting of the class file transformation. It’s also the routine that slogs through classes looking for the marker annotations that slate a class for bytecode engineering.

Getting the Agent class to execute requires a bit of legwork. First at compile time, we must create a Manifest file that specifies a premain class. The easiest way I’ve found to accomplish this is in the Ant script that assembles the jar. Here’s a snippet of the Ant task that simply stipulates the correct name-value that gets injected into the Manifest file and will cause the JVM to call my agent class prior to calling main().

```

<jar jarfile="${dist}/j105.jar" update="true" >
    <fileset dir="${classes}"/>
    <fileset dir="{configure}"/>
    <manifest>
        <attribute name="Premain-Class" value="Agent"/>
    </manifest>
</jar>

```

Ok, we’re almost done with the Agent installation. The only thing left is to pass the correct information into the JVM that causes the Agent to get called and the class-file transformer to be instantiated and installed. Here’s the command line argument that accomplishes this:

```
java -cp %CP% -javaagent:%DIST%\j105.jar=MyTransformer Main
```

It may not be clear as to what is happening here. The “-javaagent:[path to jar]\[jar file]=[String Arg to Agent]” works like this: The jar file that’s specified is the jar file that contains YOUR agent (in our case our “Agent” class). Recall that this jar file MUST contain the manifest file that specifies the premain class that is to be called. It was my experience developing this framework that simple mistakes in this setup cause nothing to happen! That is, your agent won’t be loaded, your classes won’t be transformed and your pre- or post-invocation methods won’t be called with your business methods. The source code included with this article contains simple println statements that should tell you what is (or isn’t) going on with the framework.

Byte-Code Engineering

Up to this point I haven’t introduced anything that isn’t JDK straight out of the box, but unfortunately that won’t get us where we want to go. So just to recap, we’ve developed the ability to mark classes that we’re interested in transforming (injecting pre- or post-logic). We’ve created an infrastructure to install a class transformer that will re-engineer our classes at load time based on runtime annotations. Up to this point we’ve been dealing with stock JDK components, but it’s time to take a little diversion.

Recall earlier that we had to install a class that implemented the Java interface “java.lang.instrumentation.ClassFileTransformer.” In the case of our little framework, the name of the class that does this is called “MyTransformer.” We’ve already seen pieces of this class in this article. Recall the snippet of code that iterated over a collection of Methods in a class looking for our marker (the BAM annotation). What was omitted in that snippet was the actual mechanics of the

class transformation. For stylistic reasons, I’m not going to include the entire class listing (please see the source code available with this article). So for a moment, imagine that we’ve intercepted a class being loaded, we’ve determined that it is annotated with our BAM metadata and that we now wish to introduce code that’s either executed before or after the method we’ve determined contains our annotation.

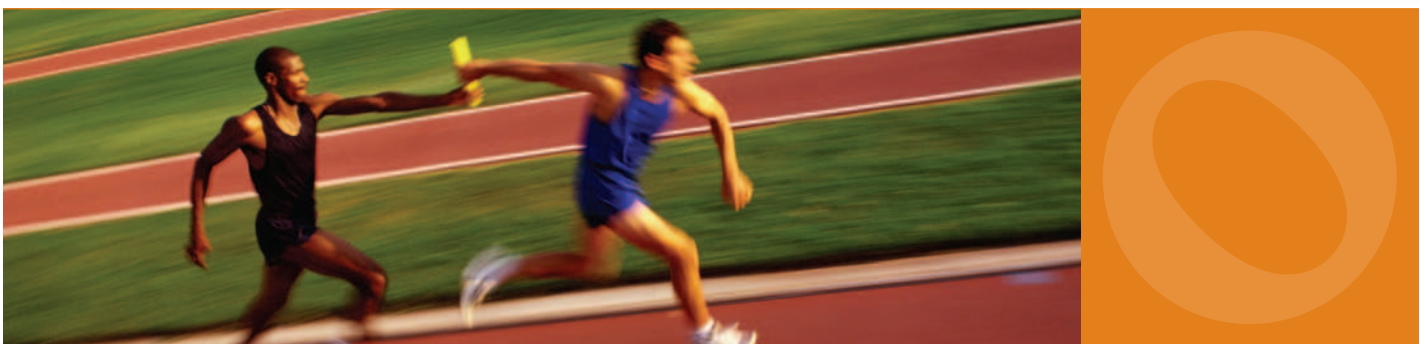
So as per the “ClassFileTransformer” interface, we must implement the following method:

```
public byte[] transform (ClassLoader cl,
                        String className,
                        Class<?> classBeingRedefined,
                        ProtectionDomain protectionDomain,
                        byte[] classfileBuffer)
{
}
```

In pseudo-code, what this method must do (in our framework) is to extract the Methods from the “classBeing-Redefined” parameter, iterate over them looking for annotations. We’ve already seen how this works. When we find a class that needs to be redefined (annotated), we must inject the pre- or post-code as directed by the annotation itself. To accomplish this injection, I used the “JavaAssist” toolkit for bytecode engineering. I’ve found this one of the easiest and most straightforward BCELs (Bytecode Engineering Libraries) to use. The code to do this looks something like this:

```
CtMethod ctM = cc.getDeclaredMethod(m.getName());
```

Grab the Xythos Baton — Easily Integrate Document and File Management into Your Next Application



The Xythos WebFile Development Suite 5.0 is the complete set of products for developers to cost-effectively integrate document and file management functionality into enterprise applications:

- **WebFile Server:** The J2EE-compatible content management platform
- **Xythos Developer Studio:** The customization toolkit and workflow designer for Xythos systems
- **Xythos Drive:** The online and offline Windows desktop collaboration client
- **Xythos Scan Client:** The solution for incorporating and managing paper documents

PLEASE VISIT US AT JAVAONE – BOOTH #610

For more information please call 1.888.4XYTHOS or visit www.xythos.com

Xythos Software • One Bush Street, Suite 600 • San Francisco, CA 94104

Xythos
software, inc.

```

if( getAnnotationProperty("insertionPoint", a.toString()).
equals("pre") )
{
    StringBuffer injectedCode = new StringBuffer();
    injectedCode.append("{");
    injectedCode.append("System.out.println(\"initialize
Spring...\");\n");
    injectedCode.append(" MyTransformer.getInjector(\
"+getAnnotationProperty("processBean",a.toString() )+"\").
executeChain(null);\n");
    injectedCode.append("}\n");
    ctM.insertBefore(injectedCode.toString());
}

```

JavaAssist makes re-engineering class files trivial. Notice that I simply concatenate together regular Java syntax and let JavaAssist translate it into bytecode and tack it on as a preamble to the method in question. Also, I've defined the method that gets attached in terms of a Spring Bean that the annotation references. This adds a convenient level of abstraction and indirection since the pre- and post-logic is defined in terms of a Spring Bean lookup key as opposed to the fully qualified class name of the code the developer wants to have executed before the business method's execution.

Bringing It All Together

Last, but not least, I've put together some classes that implement what amounts to an interceptor pattern in our annotations-based aspect system. These classes are mainly for convenience and allow us to chain together pre- and post-method interceptors. They also showcase the generic and type-safe collections supported by Java 5.0. The following snippet is the declaration for an abstract base type called "BaseInjector":

```

public abstract class BaseInjector
{
    LinkedList <BaseInjector> injectors = new LinkedList ();
    public BaseInjector ( )
    {
        System.out.println("adding me");
        injectors.add(this);
    }

    public BaseInjector( BaseInjector bi )
    {
        this();
        injectors.add(bi);
    }
}

```

```

public void executeChain(Object [] parameters )
{
    for( BaseInjector bi: injectors )
    {
        bi.execute(parameters);
    }
}

public abstract void execute(Object [] parameters);
} // class BaseInjector

```

"BaseInjector" holds a type-safe linked list of "BaseInjector" types. Recall the class that does the bytecode engineering and introduces method calls either before or after the annotated method. The code that's introduced during bytecode engineering is based on the "BaseInjector" class and is resolved at runtime via the Spring framework. So from a developer's perspective to introduce code that's executed either before a business method or after a business method, he or she would:

- Create a class (or classes) that extend "BaseInjector" and implement the execute() method.
- Annotate the business methods you want instrumented specifying two parameters, the bean lookup key that Spring will use to build your interceptor and the 'pre' or 'post' specification that tells the bytecode engineering code where to tack this interceptor on.
- Configure your Spring file.
- Run the application with the appropriate command-line switches that properly run the agent prior to main being called and installs the class-file transformer.

For completeness, here's a snippet of the Spring configuration file that wires up the interceptor. Please note that it's possible to chain together interceptors; Spring will accommodate this quite nicely via its ability to reference other beans under its control and apply them in setters or constructor arguments. Please consult Spring's documentation for more information on this. In this context, the annotation would specify the "nullInjector" as the processBean attribute and either "pre" or "post" for the insertionPoint attribute that designates where the developer intends to have the code executed.

```

<!-- Null Injector -->
<bean id="nullInjector" class="injectors.NullInjector"
singleton="false">
    <constructor-arg><ref bean="interestingInjector"/></constructor-
arg>
</bean>

```

“The annotation framework shows great promise for tool and framework writers; EJB 3.0 threatens to lean heavily on annotations and may cause the pendulum of ‘XML configuration hell’ to swing the other way”

At runtime, the injected code looks up the interceptor logic via Spring based on the value of the “processBean” attribute specified on the annotation being processed. Once retrieved, the injected code calls the “execute()” method on the bean that in turn executes whatever code happens to be in the “execute()” method. This pattern is fairly common in application server environments whereby the container must execute a series of interceptors either before or after the target business method. For instance, a J2EE container may do this on an EJB call to facilitate security, logging, statistics, and the like before actually executing the target method on the bean itself.

Summary, Conclusion, and the Ubiquitous Product Disclaimer

There are some really notable inclusions in the latest release of Java. At the very minimum, developers should embrace generics as a way of making code safer and more readable. Generics also support a tighter contract between caller and service that’s generally always a good thing. The annotation framework shows great promise for tool and framework writers. EJB 3.0 threatens to lean heavily on annotations and I suspect this may cause the pendulum of “xml configuration hell” to swing the other way. Developers will be able to introduce metadata closer to the source, which will eliminate the proliferation of deployment and runtime files that seem to be on the increase. Sun also seems to be inventing ways to open the JVM up a little to framework authors and tool build-

ers; as evidence we’ve looked at some of the new hooks available to us for class loading and on-the-fly class manipulation. I encourage developers to poke around the “java.lang.annotations,” the “java.lang.instrument,” and the “java.lang.management” packages. It’s this “poking around” that inspired some of the goodies in this article and the tiny framework that subsequently developed.

I also encourage you to download, look at, hack, and run the source code available with this article. I’ve included pertinent snips of code, but the gist of this framework is best realized by looking at the entire landscape.

Whoops, I almost forgot the product disclaimer. I wanted to make sure that I stated that this framework is a conceptual work and that I do not represent it as a full-blown aspect-oriented system. I feel the concepts here are viable and could be developed into a very rich, robust system but as set forth here are primarily intended as a backdrop to the new capabilities of Java. ☺

Resources

- <http://java.sun.com/j2se/1.5.0/docs/api/>
- <http://www.onjava.com/pub/a/onjava/2004/04/21/declarative.html>
- <http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html>
- <http://www.springframework.org/docs/reference/index.html>

What is the easiest way to create mission-critical mobile messaging applications in Java and J2EE?

By using NCL’s Provato™ !

Used by Operators, Aggregators, Content Providers and Wireless Application Developers to create and deploy mobile applications, **Provato™** gets the message across.

Provato connects with the message centers (SMSCs and MMSCs) of operators and aggregators over industry standard protocols. Developed in 100% java, it handles routing, concurrent applications, multiple networks, message persistence, failover, connection management and integrates with network management. With support for SMS, long messaging, Unicode, binary, WAP Push and Multimedia Messaging, NCL’s Provato is endorsed by operators and aggregators worldwide. Provato runs as a standalone server or deploys on a J2EE application server. Application interfaces include JMS, SOAP/XML and HTTP. A user-friendly browser-based interface simplifies (remote) management and configuration - your java application need only call *send* and *receive*. So now you can focus on the business logic of your messaging application.

For more details visit www.nclt.com/jdj
NCL Technologies Limited
Tel: +353 1 6761144 Email: jdj@nclt.com





Joe Winchester
Desktop Java Editor

The Return of the Client

Witnessed a recent BOF conversation in which the general feeling was that the browser GUI and its accompanying plethora of back-end frameworks had let people down by delivering a poor return on investment and a weak user-interface experience.

The Revenge of the Server

To predict the future you must study the past. Once upon a time servers ruled the kingdom of IT, with mainframe boxes delivering computing power to users who attached themselves via terminals. Then the GUI arrived along with cheaper and faster personal computers that promised a high-function user interface, challenging the era of the server. Early attempts to solve the client/server dichotomy exposed the weaknesses and strengths of both: the GUI delivered usability and high-end functionality while the server offered easier deployment and overall systems integrity. The server attacked the perceived strength of the client by offering a browser that provided a GUI, allowing green terminal users to put a fresh coat of paint on their applications. To all but the most belligerent among us, the browser had become the de facto user interface of the late twentieth and early twenty-first centuries.

The Clone Wars

It became clear, however, that to deliver a serious business application, an HTML user interface was just part of the story, and issues such as session state, transactions, and page management needed tackling. Back-end frameworks arose to handle these while front-end innovation faltered as forays into the world of JavaScript and other tricks to spruce up the user interface failed because proprietary extensions targeting particular operating systems couldn't do so without sacrificing the ubiquity of raw HTML. All Web applications ended up looking a lot like each other as the force pushed them toward the lowest common denominator, resulting in effective stagnation of the browser user interface.

Navigating, validating, or dealing with multiple forms simultaneously are a client applications' home territory; however, as the browser struggled with these, Mark Andreessen, Netscape's inventor, famously remarked:

There was one feature that was temporary in Mosaic: the Back and Forward buttons. That never made a lot of sense to us. Back to what? Forward to what? We thought there would be a better way to navigate. But no one ever came up with one. ... There's nothing emerging right now. Creativity stopped in 1997...
<http://www.pcworld.com/news/article/0,aid,110156,00.asp>

Form entry pages where information is left incomplete have to incur an entire server round-trip to determine the errant field, after which a new page is presented to the user with red text highlighting the areas that need attention. Sensitive information such as passwords is often not included in the error page returned to the user, so fixing the first error can just reveal further problems as this is now the missing required data. Pressing the back button can throw up the error message: "The page you requested cannot be displayed as it is the result of a post operation – please press Reload to refresh", after which a further dialog asks you to confirm whether this is really your intention. As Dr. Jakob Nielsen observed: "Billions of dollars are wasted every year in lost productivity as people wait for Web pages to perform duties that could have been handled better by a 1984 Macintosh-style graphical user interface application (www.useit.com/jakob)."

New Hope

It's never enough to score a victory simply by recognizing a strength in your opponent's argument and turning this to your advantage, for soon they will do likewise. The strength of the server is that it provides a single way to administer the client applications and HTML is a ubiquitous GUI format that can be read on

almost every operating system. The force pushed the client to invent a way to have a full-fledged user interface that could be updated centrally by the server. Java Web Start now offers this. Although applets suffered problems in their initial incarnations, improvements made in 1.4 and beyond now mean they are a viable way to deliver a fully functioning Java GUI via a browser. Java's mantra of "write once, run anywhere" suits those wishing to write a GUI that can be delivered across a wide variety of platforms, and Swing improvements in the past few years now mean that it offers a high function point that couples good platform fidelity with a set of powerful frameworks. For those wishing to program in Java and enjoy native platform widgets directly, SWT provides a very good toolkit.

The Client Strikes Back

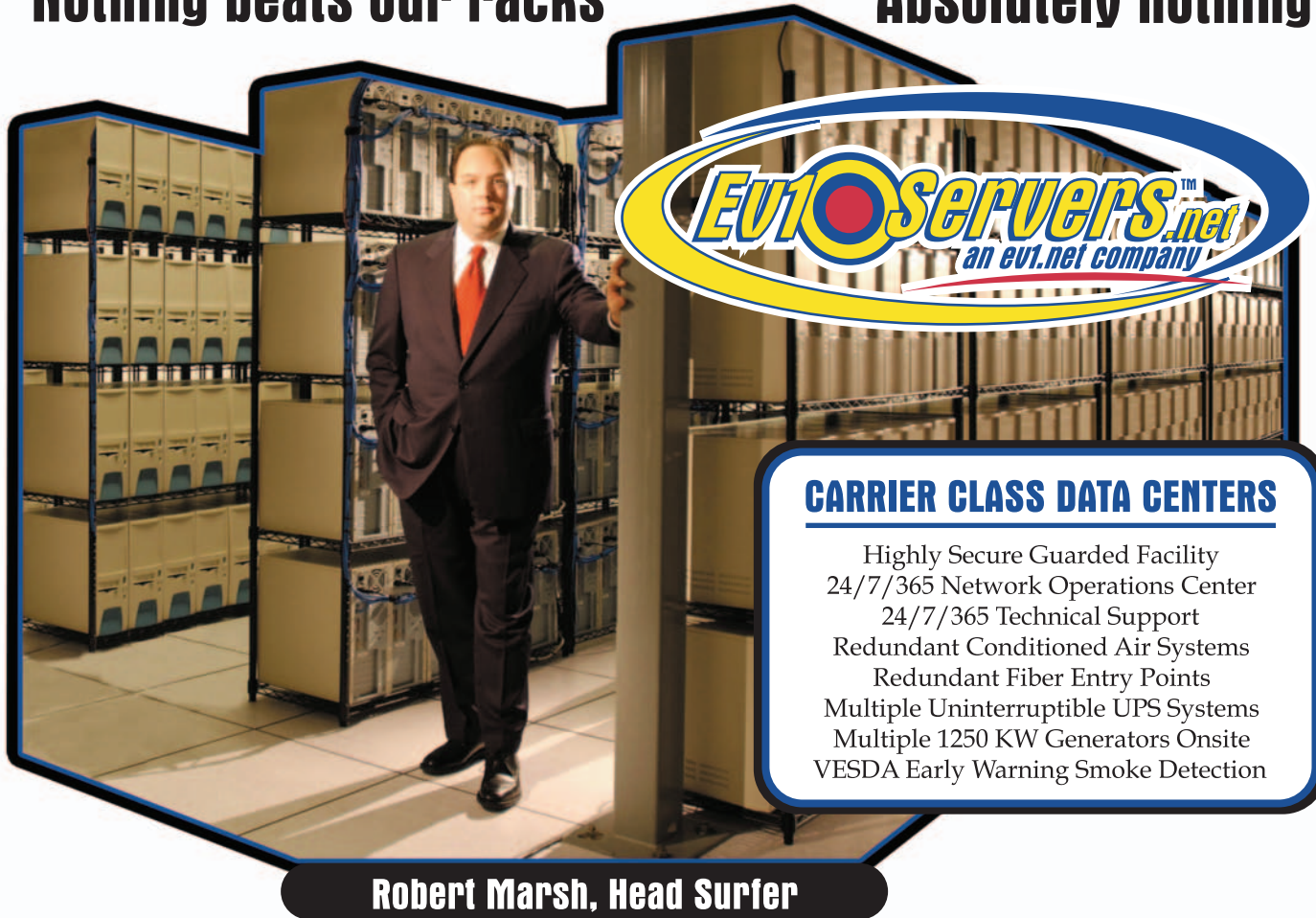
The problems that plagued early client/server for deployment and homogeneity with native appearance are solved. As the Web folks realized once HTML's glitz wore off, more is required than just serving up a fancy GUI. Application frameworks are required and JDNC is a powerful technology that provides a way to attach a Java GUI to back-end data. JSR 273, which was recently approved, promises to overhaul the JavaBeans component model specification and allow tools builders to create an easier and more powerful programming experience. Web services were once described to me as "HTML for non-carbon based life forms," the inference being that programs could send and receive messages using the simplicity of HTTP and the Web without needing a browser to render the data for a user to interpret. As more APIs from companies whose fame arose through the browser become published as Web services, the wealth of data behind the Web can become consumed by fully functioning Java client GUIs. It seems ironic that the transport of data over HTTP, the backbone that launched the Web, might become the new model for client/server, and Web services will propel the next generation of rich GUIs. The force is with us yet. ☺

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

joewinchester@sys-con.com

Nothing beats our racks

Absolutely nothing



CARRIER CLASS DATA CENTERS

- Highly Secure Guarded Facility
- 24/7/365 Network Operations Center
- 24/7/365 Technical Support
- Redundant Conditioned Air Systems
- Redundant Fiber Entry Points
- Multiple Uninterruptible UPS Systems
- Multiple 1250 KW Generators Onsite
- VESDA Early Warning Smoke Detection

Robert Marsh, Head Surfer

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from **\$299**  **Dedicated Server**

Dual Xeon 2.4 GHz
2 GB RAM • 2 x 73 GB SCSI HD
Remote Console • Remote Reboot
2000 GB Monthly Transfer Included

Instant Activation!

Over 20,000 Servers!

1-800-504-SURF | ev1servers.net

PLESK7
RELOADED
 Preferred Control Panel

IP Compliant. Price subject to change. Quantities Limited.
 *Per month. Set-Up fees apply. See web site for complete details.

Properties Editor Framework

by Swaminathan Natarajan, David Bismut, and Krishnakumar Pooloth

Solving the problem of managing application properties

Property files are frequently used in systems built using Java whether it's a thick Java client, a servlet, or a business component. Java specifies the format for a property file and provides the Properties class to read from and write to these files. However, Java is silent on the aspects related to validations of a value entered in a property file, providing room for errors to creep into an application system. How many times have you started to debug a failure in an application only to realize that it's because of an incorrect value in a property file?

The Properties Editor Framework detailed in this article was developed to solve the problem of managing application properties. It provides a convenient way to define a data type for each property and have a companion component in Swing to edit the property value. The framework can be used as a standalone tool or as part of a user interface.

One of the framework's design criteria was that it should work out of a property file, i.e., no other files or databases should be required for managing the property file. This is achieved by using certain meta-attributes to describe the characteristics of each property in the properties file. These meta-attributes are embedded along with each property in the property file as comments (Any line, in a property file, starting with '#' is considered a comment in Java).

In this article, we'll describe the framework's use, the concept of meta-attributes, and the default set of the attributes supported by the framework. And then, we'll explain the framework's design and how to extend it to support a new property data type.

Using the Property Editor Framework

The first step in using the Property Editor Framework is to decide on the Parameter type of each property in the properties file. Currently the framework supports the types listed in Table 1.

The next step is to modify the properties file and add the necessary property meta-attributes for each property. Listing 1 shows a sample properties file with meta-attributes.

The final step is to set up the environment for the Property Editor Framework and invoke the Property Editor Dialog to start updating the property values. Figure 1 shows the properties as rendered in the Property editor Dialog.

The meta-attribute *EDITABLE* indicates whether the user can edit the value of a property. In the Property Editor Dialog, only properties that have a true value for *EDITABLE* meta-attribute are displayed. This lets the application developer make certain attributes in a property file not editable through the dialog.

The *DOCUMENTATION* meta-attribute specifies the text (which can be specified using HTML) that



Swaminathan Natarajan is a technical architect with Infosys Technologies. His area of expertise is Java, repository technologies, and metadata management.

swaminathan_n01@infosys.com



David Bismut is a third-year student at the École des Mines in Nantes (EMN), a French engineering school. He's specializing in information management technologies. He was part of InStep, Infosys's global internship program in 2004.

david.bismut@gmail.com

Meta-Attributes Explained

The basic meta-attributes specified by the framework for each property are Parameter Type, Editable, and Documentation. The *Parameter Type* represents the type of a property. It's used to render a suitable component for editing the property value. The table below lists the components rendered for each of the possible values of PARAMETER TYPE.



Figure 1 Property editor dialog

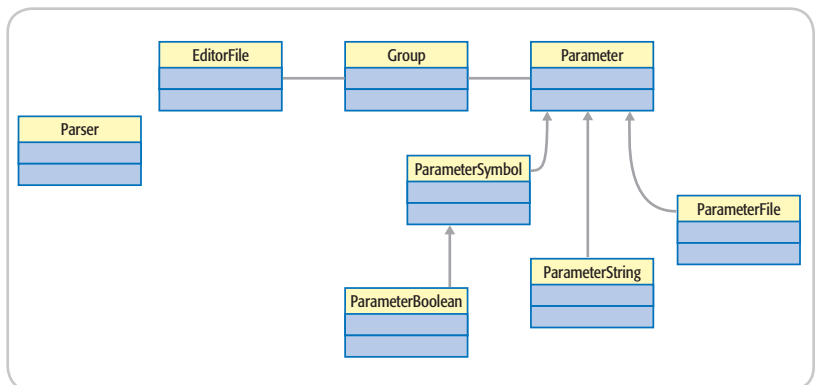


Figure 2 Class Diagram: Property Editor Framework Model

provides assistance to the end user while editing any particular property. In Figure 1, this documentation is displayed below the JTable containing the editable properties. As HTML tags are accepted, the <A> anchor tag can be used to provide a link to more comprehensive documentation.

Apart from the basic meta-attributes, there are two special meta-attributes. First one, *GROUP NAME*, is used to group related properties together. In the Property Editor Dialog, each property group is displayed in a separate tab. This could be used to group your attributes into, say, 'Basic' and 'Advanced.' Note that this parameter shouldn't be specified for each property separately. Once a GROUP NAME tag appears, all the parameters till the next GROUP NAME tag are grouped together. Another special meta-attribute is *ALLOWED VALUES*, which is used in conjunction with the SYMBOL parameter type. This lets you specify the list of valid choices for a SYMBOL property.

Parameter Type	Description
STRING	The property value can be any string
SYMBOL	The property value can be one of a set of choices. The framework allows for specifying the acceptable values for this data type.
BOOLEAN	The property value should be true or false. To be used with properties boolean in nature such as 'enableLogging'
FILE	The property value should be the full path to any file
DIRECTORY	The property value should be the full path to any directory

Table 1 Supported property types

Note: If there's a property that doesn't fit into these types, it can be set as STRING or the framework can be extended to support the new property type.

ty. These values should be separated by commas.

Adding Meta-Attributes

Now that we have a good idea about meta-attributes, let's walk through the steps in adding them to the Property File. Consider the following set of Properties.

```
LoggingEnabled
LoggingLevel
LogFile
DatabaseURI
DatabaseUser
DatabasePassword
```

Step 1: Decide the Grouping of the Properties

In the example the properties can be split into two groups based on their purpose, Logging and Database connection. Let's call these groups Logging and Database.

```
#GROUPNAME = Database
# GROUPNAME = Logging
```

Step 2: Decide on the Type for Each Property

In the example above, the types to be assigned to each property are shown in Table 3.



Krishnakumar Pooloth is a senior technical architect with Infosys Technologies. His areas of expertise include object design, component technology, Java, and expert systems. He holds a bachelor's degree in electronics and communication from Calicut University, India.

krishnakumar@infosys.com



DynamicPDF™ Merger v3.0

Our flexible and highly efficient class library for manipulating and adding new content to existing PDFs is available natively for Java

- ♦ Intuitive object model
- ♦ PDF Manipulation (Merging & Splitting)
- ♦ Document Stamping
- ♦ Page placing, rotating, scaling and clipping
- ♦ Form-filling, merging and flattening
- ♦ Personalizing Content
- ♦ Use existing PDF pages as templates
- ♦ Seamless integration with the Generator API

DynamicPDF™ Generator v3.0

Our popular and highly efficient class library for real time PDF creation is available natively for Java

- ♦ Intuitive object model
- ♦ Unicode and CJK font support
- ♦ Font embedding and subsetting
- ♦ PDF encryption ♦ 18 bar code symbologies
- ♦ Custom page element API ♦ HTML text formatting
- ♦ Flexible document templating

Try our free Community Edition!



DynamicPDF™ components will revolutionize the way your enterprise applications and websites handle printable output. DynamicPDF™ is available natively for Java.



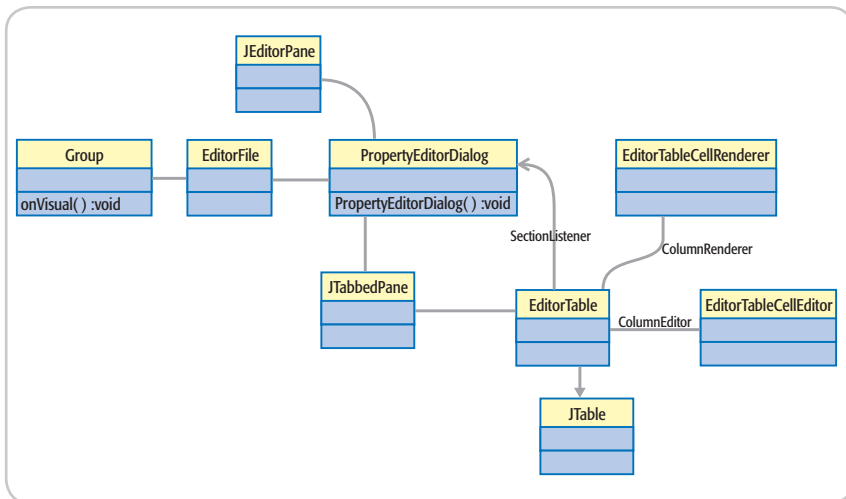


Figure 3 Class Diagram: Property Editor Framework User Interface

Parameter Type	Edit Control
STRING	Text Control
SYMBOL	Combo box (shows the list of allowed values)
BOOLEAN	Combo box (with True or False)*
DIRECTORY	Text control + Dir selector
FILE	Text control + File selector

Table 2 Property types and edit controls

Property	Property Type
LoggingEnabled	BOOLEAN
LoggingLevel	SYMBOL (debug warning error)
LogFile	FILE
DatabaseURL	STRING
DatabaseUser	STRING
DatabasePassword	STRING

Table 3 Properties and the corresponding data types for the sample case

Property Type	Control
STRING	FormattedTextField
SYMBOL	FormattedComboBox
BOOLEAN	FormattedComboBox (With just True and False)
FILE	FormattedFileChooser

Table 4 Property types and corresponding visual components

Step 3: Add Meta-Attributes to Each Property

The complete listing of this sample is available for download along with the source (see Listing 2).

Design of the Property Editor Framework

In this section, we will detail the design of the Properties Editor

Framework. To reduce complexity, we'll break the discussion into: Creating the model of the Properties Editor component, creating the UI of the Properties Editor component, and the framework's persistence mechanism for the properties file.

Creating the Model

The classes involved in reading the properties file into memory and creating the data model are depicted in Figure 2.

The EditorFile class, which extends the File class, is used as a thin wrapper for the Properties file. The EditorFile class provides the facility to add an Observer, which will be notified when changes are made to the property values.

The Parser class is a simple line parser that reads data from the EditorFile instance line-by-line and creates an in-memory model of the property file. It also defines the regular expression patterns to match the different meta-attributes supported by the framework. This class creates a new instance of Group class each time the meta-attribute '#GROUP NAME = <Group Name>' is encountered.

For each property key-value that falls under the same #GROUP NAME, an instance of one of the sub-classes of the Parameter class hierarchy is created. The exact sub-class is decided by the value of the meta-attribute #PARAMETER TYPE, which also keeps track of the value of meta-at-

tribute EDITABLE that's used later by the view to decide whether to make it visible.

Creating the User Interface

The classes involved in rendering the user interface of the component are depicted Figure 3.

Referring to Figure 1, the visual area of the dialog is split between the Panel used to render the table of property key-value pairs and the JEditorPane showing the document-ation text of different properties.

The PropertyEditorDialog class is responsible for a big part of the work involved in creating the user interface for the component. As the name suggests, it extends JDialog; it also implements the Observer and ListSelectionListener interfaces. The Observer interface is implemented to change the title of the dialog when any of the property value is changed.

The PropertyEditorDialog obtains the number of groups of property key-values in the EditorFile and creates an equal number of JTables to render them. These JTables are, in turn, put in a JTabbedPane, one tab for each group, the tab's title corresponding to the group name. The PropertyEditorDialog attaches itself as the ListSelectionListeners for all the Jtables. The PropertyEditorDialog gets the description text for the property on the valueChanged event and displays it on the description JEditorPane.

Finally, let's look at the rendering and editing mechanisms for the Property Key-Value pairs.

The toVisual method of the Group creates the EditorTable and the TableModel for the EditorTable. The EditorTable, in turn, creates and sets an instance of the EditorTableCellRenderer as the cell renderer and an instance of the EditorTableCellEditor as the cell editor for the columns of the EditorTable.

The EditorTableCellRenderer and EditorTableCellEditor create the controls necessary to render and edit the property values. The different controls for each Property type are listed in Table 4.

EditorTableCellEditor adds an anonymous inner class to each control overriding the functionality for updating the controls to include actions to set the new value back to the model and invalidate the model.

Persisting the Properties

Writing the Property key-values back to the Properties File is pretty straightforward. A BufferedWriter is created with the instance of EditorFile as the FileWriter. The Parameters are written group-by-group into this file. Concatenating the text form of each instance of the Parameter hierarchy creates each group.

Extending the Framework To Add New Type and Controls

Now we'll discuss adding a new type for a property and the corresponding control to edit the property's value. Say your application uses JDBC to connect to a database and you want to give the user the ability to specify

the JDBC driver to use via a property. The framework, by default, doesn't support the required type, which lets you choose a valid driver from a list. We'll see how to extend the framework to add this new type.

To display all the JDBC drivers we could use a dropdown list with all the JDBC drivers in the classpath. Let's name the new parameter type JDBC_DRIVER.

For the new type, we'd need to define a new class on the model side, say, ParameterJDBC_Driver, which is a sub-class of the ParameterSymbol class. The new class needs to implement the method "toText" that should return the driver's fully qualified class name. The following code snippet does just that:

```
public class ParameterJDBC_Driver extends
ParameterSymbol {
String strBaseClassName;
...
public ParameterJDBC_Driver(String name,
String value, String baseClassName, bool-
```

```
ean editable, String doc) {
Driver[] driversInClasspath = ClassUtils.
findAllDriversInClasspath(baseClassName);
...
setAllowedValues(driverNames);
type = "JDBC_DRIVER";
}
public String toText(){
String newLine = System.getProperty("line.
separator");
return toTextType() + newLine +
toTextValue();
}
}
```

Now we need to add the following code to the createParameter method of the Parser class. It creates an instance of ParameterJDBC_Driver when the Parameter type "JDBC_DRIVER" is encountered in the properties file.

```
if (pType.compareToIgnoreCase("JDBC_DRIVER")
== 0) {
p = new ParameterJDBC_Driver(pName, pValue,
pBaseClass,pEdit, pDoc);
}
```

The ODTUG Conference Is Jazzed!

Business Intelligence • Data Warehousing • Methodology • Development DBA
Professional Development • Application Development (J2EE, JDeveloper, PL/SQL, and HTML DB)

- NEW! Best Practices in Software Architecture Symposium
- NEW! Hands-On Training
- Oracle Product Updates
- Vendor Presentations
- Tool Topics (3-hr. in-depth sessions)
- 100+ Technical Sessions
- Networking
- New Orleans Jazz

For Sponsorship Opportunities
Call 910.452.7444
www.odtug.com

Save \$100
Register by May 30

ODTUG NOW 2005
June 18-22
Sheraton New Orleans Hotel
NEW ORLEANS, LOUISIANA

Reserve your hotel room online at www.odtug.com/2005_conference_location.htm or call 888-627-7033 or 504-525-2500 by May 30 and reference ODTUG for the special rate of \$159/night. A limited number of rooms are available at the government rate on a first-come basis. Artwork by Ivey Hayes, North Carolina Artist www.mccluregallery.net

“How many times have you started to debug a failure in an application only to realize that it’s because of an incorrect value in a property file?”

This completes the changes needed on the view front, since we have decided to show the values in a drop down list, we can reuse the combo box control used to display values of type Symbol/Boolean. But we have to make changes in a couple of classes to ensure that the dropdown list is correctly rendered for the new Property type.

The following code needs to be added to the createComponent method of the EditorTableCellRenderer class. It will render the FormattedComboBox control for the new Property type.

```
else if (cellValue instanceof
ParameterJDBCDriver) {
    o = new FormattedComboBox(cellValue,
((ParameterJDBCDriver) cellValue).getAllowedValues(), value.toString()) {
        public void itemStateChanged(ItemEvent e)
        {
        }
    };
}
```

Now we need to add the following code to the createComponent method of EditorTableCellEditor class to ensure that editing changes are reflected correctly.

```
else if (cellValue instanceof
ParameterJDBCDriver) {
    o = new FormattedComboBox(cellValue,
((ParameterJDBCDriver) cellValue).getAllowedValues(), value.toString()) {
        public void itemStateChanged(ItemEvent e)
        {
            ((ParameterJDBCDriver) cellValue).setValue(
getSelectedItem().toString());
            PropEditor.eFile.setModified(true);
        }
    };
}
```

The last step is to compile all the changes and we’re good to go! You can fire up the PropertyEditor Dialog and verify that it works.

Adding a New Meta-Attribute

In some cases, you might need to add one or more custom attributes to handle a new type properly. Say, in the previous example, we wanted to build the ability to store the name of the base class (for the JDBC driver this would be java.sql.Driver) whose subclasses we want to select from. It would make the new control more generic. To implement it, we would need to add a new meta-attribute specific to this type, say, BASECLASS. To enable the framework to parse this new meta-attribute, you’d need to add a new regular expression pattern to the Parser class that matches the newly defined meta-attribute and then write the required processing so this data is passed to the constructor of the corresponding Parameter class.

Conclusion

Property files are frequently used in Java-based applications so their execution can be controlled externally. In this article we described the design and use of a lightweight framework for managing these property files. The framework is easily adoptable and allows for extensions. However, we believe it can be made even better. Java 1.5 supports the XML format for specifying properties. Enabling this framework support would ease the task of specifying documentation attributes (currently documentation need to be specified in a single line). We hope you enjoy working with this framework as much as we did developing it. ☺

References

- <http://java.sun.com/j2se/1.5.0/docs/api/java/util/Properties.html> - Read the documentation about the Properties class. It also specifies the format in which property file needs to be specified.

- <http://java.sun.com/docs/books/tutorial/uiswing/components/tabbedPane.html> - Read the Java tutorial on how to use JTabbedPane.
- <http://java.sun.com/docs/books/tutorial/uiswing/components/table.html> - Read the Java tutorial on how to use JTable.
- <http://www-106.ibm.com/developerworks/java/library/j-tiget02254.html> - A more comprehensive read about the property file XML format introduced in Tiger.
- <http://www.javaworld.com/javaworld/jv-atip135.html> - This article discusses a layered approach so that a large property file can be split into manageable smaller ones. It also introduces a tool helpful in managing these layered property files.
- <http://java.sun.com/docs/books/tutorial/extra/regex> - Read the Java tutorial on how to work with regular expressions.

Listing 1: Properties file with meta-attributes

```
#GROUP NAME = General
#PARAMETER TYPE = DIRECTORY
#EDITABLE = true
#DOCUMENTATION = Help for directory
directory_param = C:\
```

Listing 2: Excerpt of the property file for the sample case

```
#GROUPNAME = Logging
#PARAMETER TYPE = BOOLEAN
#EDITABLE = true
#DOCUMENTATION = Decides whether to
log messages when application runs
LoggingEnabled = true
...
#GROUPNAME = Database
#PARAMETER TYPE = STRING
#EDITABLE = true
#DOCUMENTATION = Help for directory
DatabaseURI = valueofdatabaseuri
```

Open Up!

Portals—Get what you want easily



company, product and service names may be trademarks or service marks of others.

Ask the Experts

Prolifics is 1 of only 3 WebSphere Service Providers retained by IBM. Open up with portals! With 26 years of industry expertise, Prolifics' certified WebSphere portal experts deliver a personalized, secure, unified view to your enterprise assets.

Our portal solutions build agility into your business — enabling you to reach new markets, achieve customer satisfaction beyond expectation, and ease collaboration within your enterprise. Our teams of specialists work together with your business experts from gathering the business requirements, translating them to a technical specification, to delivering the final production application.

Portal Executive Assessment

Prolifics experts help you articulate the value of portals to your business sponsors and perform a quick business review identifying portal benefits as it applies to your organization.

Portal Workshops and Proof of Concepts

Prolifics demonstrates valuable portal features and translates the breadth of your portal objectives into a step-by-step roadmap to meet your current and future needs.

Portal Jump Start

Prolifics' consultants work with you from the start to install, configure and jump start your team on using WebSphere Portal Software.

Portal Development and Mentoring

Prolifics' innovative approach to building portals drives down your overall cost and transforms the way you conduct business. We mentor and arm your team with the skills to manage, maintain and extend the portal environment.

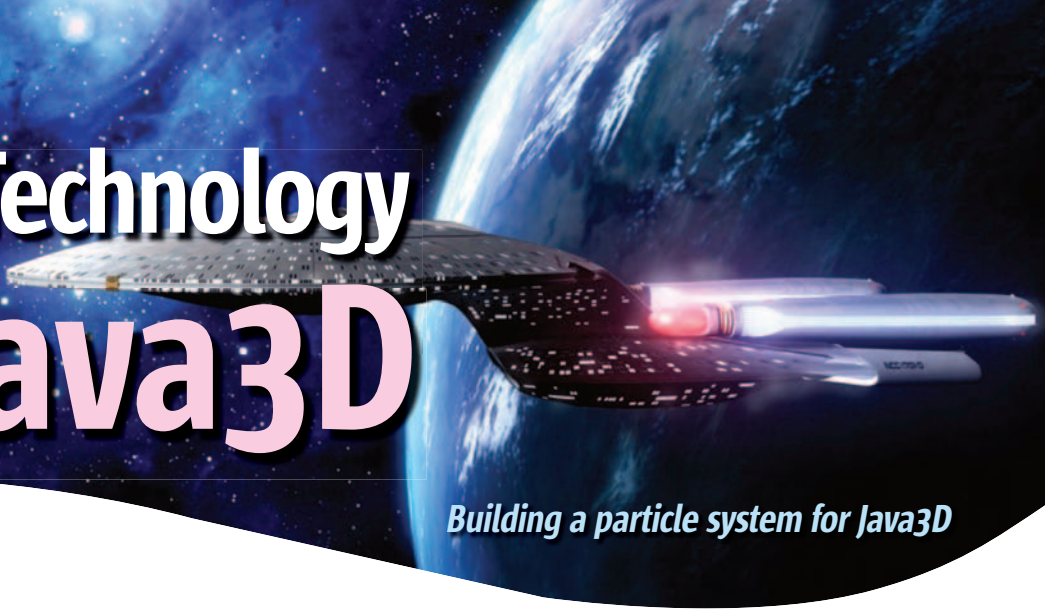
Prolifics

Winner of 2003 IBM BUSINESS PARTNER LEADERSHIP AWARD

116 John Street, New York, New York 10038
800.675.5419
solutions4websphere@prolifics.com
www.prolifics.com

Prolifics 

Star Trek Technology for Java3D



Building a particle system for Java3D

by Mike Jacobs

The Star Trek universe has inspired many technology ideas but I'm disappointed I don't have a transporter yet. One Star Trek technology that has been available for sometime is the particle system. No, this is not an exotic propulsion system for your flying car. The particle system was invented to animate the Genesis effect in Star Trek II: The Wrath of Khan. While the Genesis device was used to transform a barren planet into one full of life, we can adopt this technology for more modest effects in Java3D.

In the Beginning

In previous articles, we've focused on creating planetary surfaces with Java3D. One challenging area of graphics programming is rendering irregular or ill-defined objects like clouds, smoke, or fireworks. William Reeves faced that challenge when Lucasfilm was asked to create a planetary creation effect called the Genesis effect for *Star Trek II: Wrath of Khan*. The idea was that a planet would be hit with a missile that would transform it from a barren wasteland into one full of life. Explosions and flames on a planetary scale gave birth to a new form of animation called a particle system.

Reeves' original paper (see references) describes a particle system as one defined by clouds of primitive particles or points in three dimensions. These particles change and move with time making a particle system dynamic. How a particle changes or moves is based on a controlled stochastic process giving it a natural look. How particles evolve in a particle system is called the particle life cycle.

Your morning shower is just like a particle system. Particles are born and emitted by the system. Where the particles are born and where they are headed is assigned by the particle system. Your plumbing system determines the water temperature and velocity of the droplets. Particles exist and change under the influence of external

forces. The room temperature and gravity affect how the water changes temperature and where it collides with you or the tub. So where does that fancy stochastic process come into play? To make the rate of particle emission, ejection angle, velocity, or any other attribute more interesting we need to vary them in slightly unpredictable ways. Reeves described the approach of adding a randomly selected variance to the central value of an attribute:

$$\text{Attribute} = \text{CentralAttributeValue} + \text{Random}() * \text{AttributeVariance}$$

This approach can be applied to just about any attribute of the particle or the particle system. Figure 1 provides a simple example. The particle system evolves over time by repeating a series of steps and varying the attributes along the way. The steps are:

1. New particles are initialized and emitted using varying attributes.
2. Particles past their life expectancy die and are removed.
3. Surviving particles are updated based on external forces, velocity, etc.
4. The particles are rendered.

This cycle is repeated until the particle system has no more particles or lives beyond its lifetime. That's all you need to know to get started building a particle system, so let's build one for Java3D.



Mike Jacobs is a mild-mannered technical architect by day and recreational programmer by night. Mike works at the Mayo Clinic. Please provide feedback and guidance for future *JDJ* articles to mnjacobs.java-developersjournal.com.

mnjacobs.javadevelopersjournal.com

Copyright 2005 © Mayo Foundation for Medical Education and Research

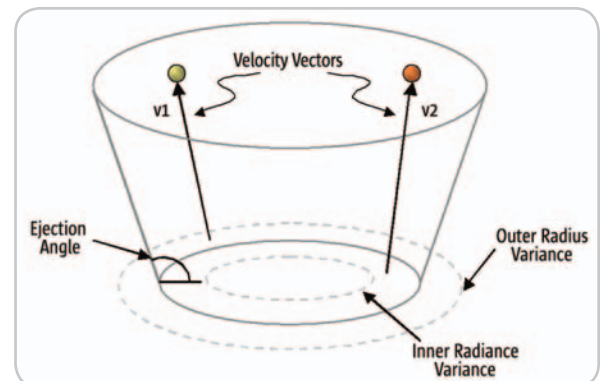


Figure 1 Particle systems emit particles based on a creation area and randomly varying attributes such as location, ejection angle, and velocity.

Transporting to Java3D

Before we build the particle system in Java3D, let's lay out the objectives. First of all, the particle system should be easy to add to the scene graph just like any other Java3D shape. We should allow the Java3D *TransformGroup* to be used to position and orient the particle system. The design should allow us to use anything from pixels to *Shade3D* objects for our particles. The particles should be emitted from a variety of nozzle shapes and be affected by external forces like wind or gravity. These objectives should give us the flexibility to graduate from simple water fountain particle systems to tornado simulations. Before we jump straight into an F5 tornado, let's get a simple spray working.

Figure 2 is a subset of a design that satisfies our particle system objectives. It's probably easiest to describe it from the bottom up so let's start with the *Particle* object. This obviously represents a particle in the particle system and logically maintains its position, velocity, and acceleration. I used the word "logically" because I am fibbing a bit to make it easier to describe. A *ParticleEmitter* object emits particles and controls the movement of the particles as you might expect. The particle emitter delegates the initial position of particles to a *GenerationShape* object. During the animation cycle, *ExternalInfluence* objects such as *Gravity* affect the particles. All of these objects are independent of Java3D so they could be used in other environments as well. Now I'm going to fess up about the particle location. The location of a particle is actually maintained by the particle emitter. This is done so that the locations of all particles can be shared in one array for use in a Java3D specific *Shape3D* implementation of a particle system.

Do You Have a Point, Spock?

So far we have satisfied just a few of our objectives. Let's use the particle emitter to make a single *Shape3D* particle system. We can do this by making *ParticleSystem* a subclass of the *Shape3D* class and using a *PointArray* for the geometry. Luckily for us, the particle emitter has a float array of particle locations that will fit nicely into a *PointArray*. This geometry class is about the simplest supported by Java3D, accepting a float value for each *x*, *y* and *z* location of the point. The geometry needs to change as the particles move so this means that this particle system should implement the *GeometryUpdater* interface. If you're not familiar with how to change the geometry of a Java3D shape during runtime, refer back to my previous article ("Casting Perlin's Movie Magic in Java3D" [JDJ, Vol. 10, issue 3]) for an overview. The last piece of the design is the *ParticleSystemManager* that is responsible for notifying the particle systems to run through their life-cycle steps. Before we dig into the details of how it works, have a look at two of the examples included in the source code and shown in Figure 3. (The source code can be downloaded from www.jdj.sys-con.com.)

The particle system is created in Listing 1. The particle emitter is created with a point generation shape having a spread angle of 45 degrees, and specific central and variance values for emission rate, initial velocity, and particle lifetime. We fade the particles by adding the *FadePoint* influence to the particle emitter (one of many influence objects included in the source code). This influence

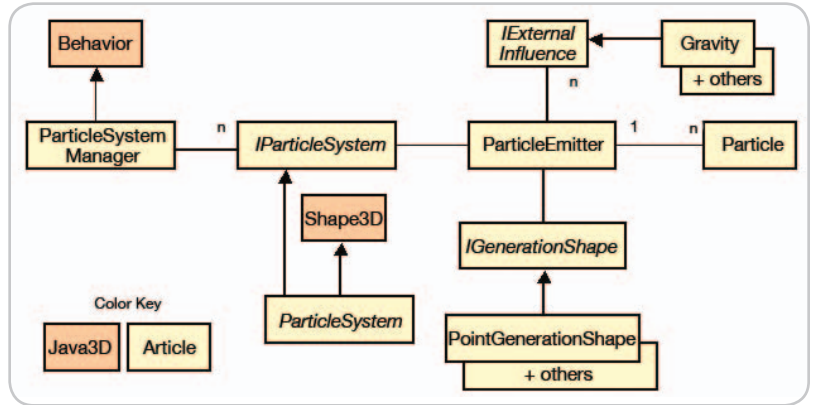


Figure 2 A partial UML design diagram of the particle system for Java3D

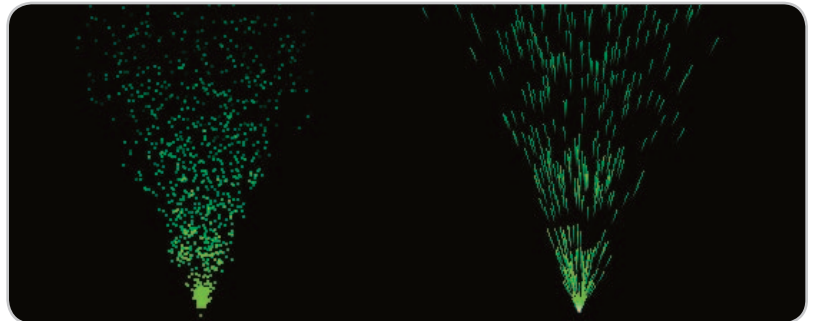


Figure 3 A point generation shape is used with a particle emitter to spray green particles. The particles on the right are emitted with motion blur.

Build Incredible Interactive Diagrams

with **JGo™**

New!
JGo for SWT/Eclipse
JGo Instruments for meters, dials, gauges

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**

Free evaluation at:
www.nwoods.com/go
 800-434-9820 or 603-886-9173

gradually changes the transparency of the particle as it ages. Now the particle system is created with the emitter and a green particle color. Adding the particle system to the scene graph is not shown due to space constraints, but it's added just like any other Java3D shape. Finally the particle system manager is added to the scene graph and the animation begins.

The *ParticleSystemManager* is a Java3D behavior that notifies all active particle systems after a specified time has elapsed. I covered behaviors and their use in animation in my previous article so refer back to it if you need a refresher. This implementation uses a combination of elapsed frames and time to create a stable animation cycle. The particle system manager starts and maintains the Reeves particle system life cycle.

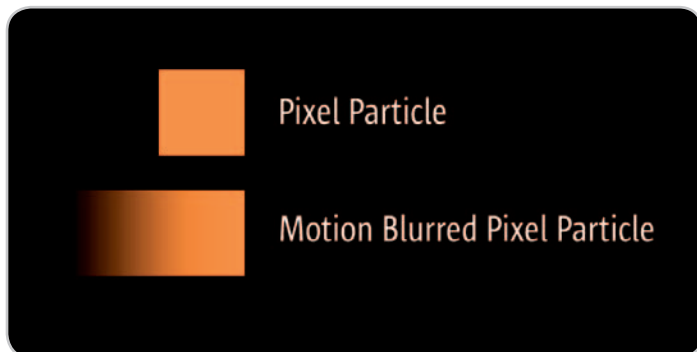


Figure 4 Point particles can be motion blurred with a line having a varying transparency. The particles are moving to the right in this figure.

Emit New Particles

The particle system manager notifies the particle system that enough time has elapsed. This time is typically less than 50 milliseconds. The emitter determines how many particles need to be initialized and emitted based on the emission rate, emission rate variance, and the amount of time since the last notification. The speed of the particles is initialized based on the velocity and velocity variance while the generation shape determines the initial location. The generation shapes in the source code include point, line, disk, and radial shapes. Particles are assigned a lifetime using the same central value plus the variance approach used for other attributes.

Bury the Dead Particles

During the previous cycle, particles were aged based on the elapsed time. Now the particles that have exceeded their lifetime are collected and recycled for future emissions.

Update the Surviving Particles

The remaining particles are aged, changed by any external influences, moved, scaled, and rotated. Aging simply increments the age of the particle based on the elapsed time. External influences can affect visual characteristics such as color or transparency, or physical attributes such as scale, acceleration, velocity, or position. Updating physical attributes does involve a bit of physics, but hopefully it hasn't been too long since your last physics class.

As part of applying the influences, the accelerations are accumulated from each influence. For example, there may be a *Gravity* and *Wind* influence affecting the particles. Gravity would obviously apply a downward acceleration while wind would apply a horizontal acceleration. These accelerations and the existing velocity and position are used to move the particle. The particle is moved in the following method:

```
public void move(float dt) {
    float[] position = getLocalPosition();
    Vector3f v = getLocalVelocity();
    Vector3f a = getLocalAcceleration();
    v.scaleAdd(dt, a, v);
    // Scale add not used with position
    // to reduce number of objects created.
    float x = position[0] + vx * dt;
    float y = position[1] + vy * dt;
    float z = position[2] + vz * dt;
    setLocalPosition(x, y, z);
}
```

This implementation uses Euler's approach to numeric integration. This approach works for us provided the time differential (*dt*) between frames is small. The *ParticleSystemManager* controls the elapsed time so we can ensure the time differential is small. Other approaches to numeric integration such as Modified Euler, Heun, or Runge-Kutta could be used if we needed more accuracy for our particle simulation.

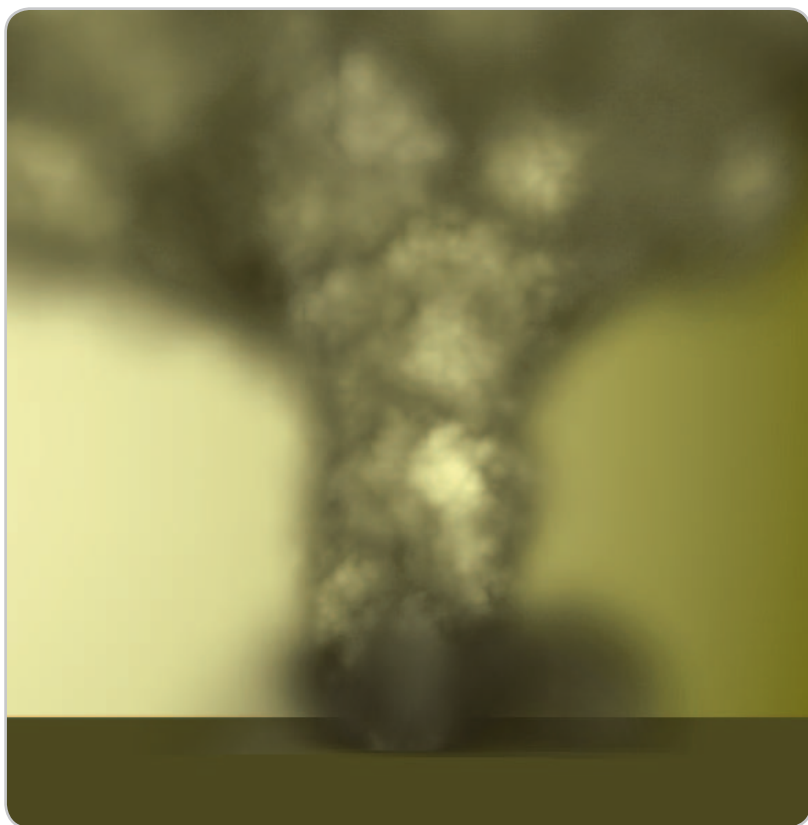


Figure 5 About 200 Shape3D particles can be used to create fantastic particle systems like this tornado.



INNOVATE

with the Power of Java™

Connect with the full power of Java™ technology at the JavaOne™ conference.

In-depth EDUCATION

Evolve your skills in 300+ of expert-led technical and Bof sessions.

Real-world INNOVATION

Evaluate proven tools and technologies in the JavaOne™ Pavilion.

Global COMMUNITY

Celebrate the tenth anniversary of Java technology.

Visionary INSIGHT

Hear what the future holds from industry leaders.

June 27–30, 2005

JavaOne™ Pavilion: June 27–29, 2005
Moscone Center, San Francisco, CA

Experience a week unlike any other

EXPERIENCE THE 10TH ANNUAL JAVAONE™ CONFERENCE.

Core Platform | Core Enterprise | Desktop | Web Tier | Tools | Mobility and Devices | Cool Stuff

Save \$100!

REGISTER

by June 27, 2005 at java.sun.com/javaone/sf.

JavaOne™

Sun's 2005 Worldwide Java Developer Conference™

Because we are currently dealing with point particles, we'll skip scaling and rotation for now since these make little sense for points.

Render the Particles

This step is trivial in Java3D because the previous step changed the location of the points representing the particles. The previous step updated the geometry of the particle system shape using the *GeometryUpdater* interface and Java3D renders the new positions for us. As

the particle system life cycle evolves, the position of the particles changes, creating the animation. Because we are dealing with points, some simulations can be disorienting unless we introduce motion blur.

It's All a Blur

You've probably seen motion blur when the Enterprise goes to light speed. Motion blur on film can occur when the subject moves quickly while the camera shutter is open. While you probably don't want the blur when taking photographs, motion blur makes animation look more lifelike and increases the perceived frame rate. How can we take our particle system to the next level and add motion blur?

Remember that each cycle of the particle system life cycle is repeated after a very short elapsed time. If we treat the elapsed time as an open camera shutter, then we want to blur the particle movement during this elapsed time. During the elapsed time, the particle moves from one location to the next. If a line segment is used instead of a point, we can connect the previous location with the new location and vary the transparency of the line to create the blur as depicted in Figure 4.

The *MotionBlurredParticleSystem* implements motion blurred points using the Java3D *LineArray* geometry array class. The *ParticleEmitter* supports both the previous and current location of the particle, enabling the particle system to use this information to create the line segments. The blurring is accomplished by assigning colors to the end points of the lines with different alpha values. The current location has a fully opaque alpha value while the previous location has a fully transparent alpha value. Java3D takes care of smoothly interpolating the transparency of the line, conveniently creating the motion blur for us.

Light Speed, Mr. Sulu!

Using points and motion-blurred lines for particles in Java3D performs very well. Several thousand particles can be used simultaneously on modest hardware with reasonable performance. I pushed my antiquated one-gigahertz machine to run three motion-blurred particle systems consisting of 8635 particles, gravity, and particle bouncing at 14 frames per second. The 14 frames per second is not stellar; it looks terrific with the blurring. Clearly the advantage of using points or lines is that they are fast and allow the use of thousands of particles. The disadvantage

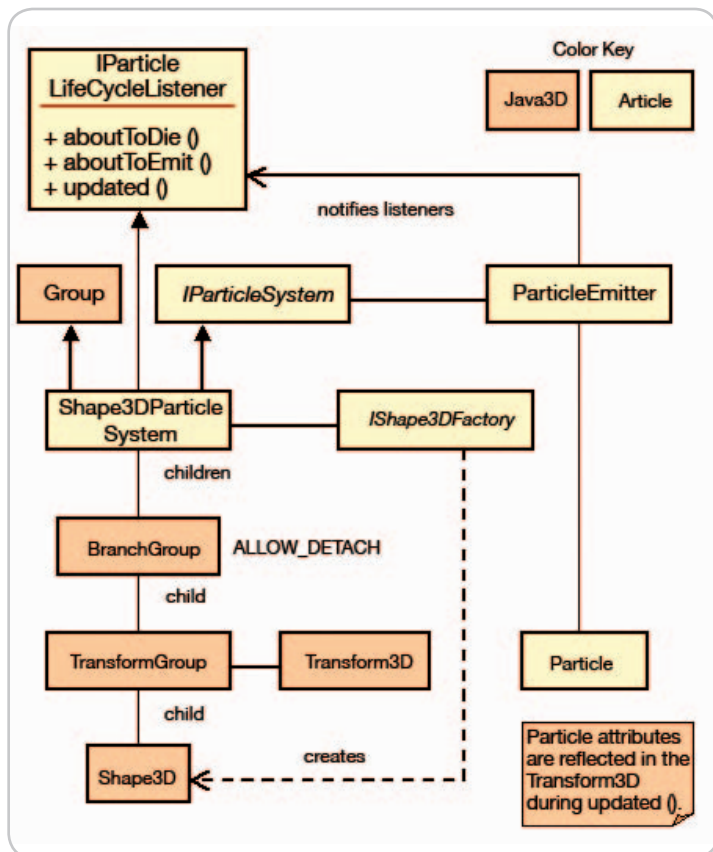


Figure 6 The UML design diagram to support shapes as particles

Orientation Conversion
 Euler (ψ, θ, ϕ) to Quaternion $[(x, y, z), w]$
 $Q_{\psi} = [(\sin(\psi/2), 0, 0), \cos(\psi/2)]$
 $Q_{\theta} = [(0, \sin(\theta/2), 0), \cos(\theta/2)]$
 $Q_{\phi} = [(0, 0, \sin(\phi/2)), \cos(\phi/2)]$
 $Q = Q_{\psi} * Q_{\theta} * Q_{\phi}$

New Orientation While Spinning
 Angular velocity $\omega = [(d\psi/dt, d\theta/dt, d\phi/dt), 0]$
 $dQ/dt = 0.5 * \omega * Q$
 For a short time interval Δt :
 $Q(t+\Delta t) = Q(t) + \Delta t * dQ/dt$
 $Q(t+\Delta t) = Q(t) + \Delta t * 0.5 * \omega * Q(t)$

Figure 7 A quaternion can be easily converted from Euler angles and is ideal for spinning objects

Pop Quiz Hot Shot

What is a *Quaternion*?

- a) A resident of the planet Quatern
- b) A rogue protein responsible for Bad Programmers Disease
- c) A mathematical representation of the rotational position of rigid bodies
- d) An elementary particle found in high energy collisions of animated characters

Answer: c. Invented by Sir William Rowan Hamilton in 1845. Quaternions are supported by Java3D, but barely documented.

to using points or lines is the lack of scale. Each particle is one pixel in size regardless of the distance from the viewer. While you can change the pixel size of the particles via the *PointAttributes* and *LineAttributes*, the particles are still all the same size. Ideally our particles should have scale and, for performance sake, reduce the need for a large number of particles. An approach to solving these issues is to make the particles full-blown Java3D shapes. With this approach we can create fantastic effects like the tornado shown in Figure 5.

We Need More Power, Scotty!

Let's extend our particle system design to incorporate *Shape3D* particles. We still want to easily add the particle system to the Java3D scene, and reuse as much of what we've done up to this point. Recall that our particle systems use an emitter to control the initial position and velocity of the particles. The point and motion blurred particle systems deal with pixel size particles so we'll have to create a new type of particle system to handle Java3D shapes. So far, the particle systems have been *Shape3D* objects, so how can particles be any *Shape3D* object? Java3D supports the aggregation of shapes into a *Group*.

As you can see from Figure 6, the *Shape3DParticleSystem* is a subclass of the Java3D *Group* class to allow shapes to be grouped together into a particle system. By implementing the *IParticleSystem* interface, the *Shape3DParticleSystem* can use the particle emitter unchanged. To help organize and control the shapes, the shape particle system maintains a scene graph segment for each shape in the particle system. Each shape has a scene graph segment consisting of a branch group to maintain membership in the particle system and a transform group to control the location, scale, and rotation of the shape.

Because particles are born and die during the particle system life cycle, shapes must be added and removed from the scene during the animation. Java3D limits the changes to the content of live scene graphs to branch groups. Provided the group (the particle system) has the *ALLOW_CHILDREN_EXTEND* and the *ALLOW_CHILDREN_WRITE* capabilities set and the branch group has the *ALLOW_DETACH* capability set, the branch group and its children can be added or removed from the scene. For our purposes, the only child of the branch group is a transform group. The transform group maintains the standard Java3D translation, scale, and rotation attributes of its child shape in a *Transform3D* object. With this structure in place, let's briefly review the Reeves life cycle for our new shape particle system.

Emit New Particles

From Figure 6 you can see that the shape particle system implements the *IParticleLifeCycleListener* interface. The particle emitter notifies listeners as particles evolve through their lifetime. This notification can be used to create additional effects such as spawning additional particle systems. Just before particles are emitted, the *aboutToEmit()* method is called on the listener, passing a list of particles to be emitted. The shape particle system

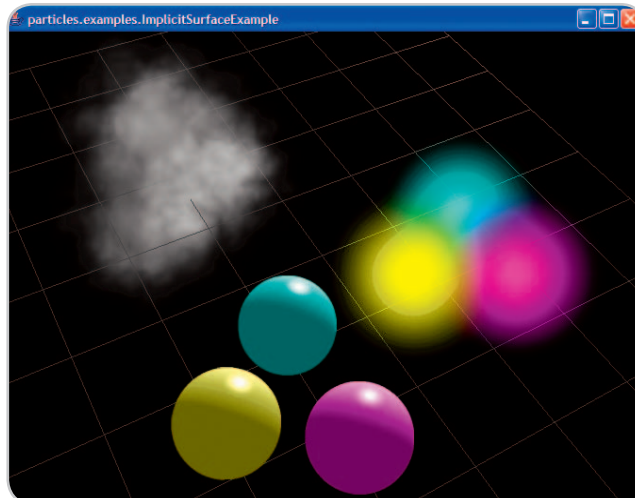


Figure 8 All objects in this scene are *OrientedShape3D* objects. Note the correct lighting on the fake spheres.

reflects the initial particle position in the transform group and adds the branch group, transform group, and shape to the scene.

Bury the Dead Particles

When the particle emitter is ready to bury dead particles, it notifies listeners of their impending demise by calling



COMMON CONTROLS

www.common-controls.com



The Java™ Presentation framework for J2EE™ Web applications

Based on:

- Java™
- Servlets™
- Java Serverpages™
- and Struts



For Free!

Get your free trial version
– www.common-controls.com
See the common controls in action
– go for the [Online Demo!](#)

Contains the most common control elements which are required for the development of J2EE™ applications with rich HTML frontends like:

Lists

Trees

Tabfolders

Menu

Forms

BreadCrumbs

Calendar

Colorpicker

www.common-controls.com

the *aboutToDie()* method. The shape particle system takes the news pretty well by removing the branch group for the particles from the particle system, removing the shapes from the scene. To reduce object creation, the shape, transform group, and branch group are recycled for a future reincarnation.

Update the Surviving Particles

While particles are alive, the particle emitter applies the influences and the particles are moved, scaled, and rotated. After the particles have been updated by the particle emitter, it notifies the listeners by calling the *updated()* method. The shape particle system reflects the new particle position, scale, and rotation in the transform group. We discussed how linear acceleration and velocity of a particle can affect its new position, but how about rotation?

Slicker Than Euler

Realistic rotation of *Shape3D* particles with Euler angles can be mathematically intensive and computationally expensive. I'll try to keep the math to a minimum but if you are interested in the details, have a look at the references. If you have read much about three-dimensional graphics, you've probably already heard of Euler angles. An example of Euler angles is the yaw, pitch, and roll used to describe the orientation of an airplane. There are a few problems with using Euler angles that make them difficult to use for animation.

The order in which Euler angle rotations are applied can result in different orientations. While applying the rotations, a degree of freedom can be lost to something called a "gimbal lock". Over the course of multiple rotations, numeric corrections are often needed to keep the rotational animation looking good. Too make matters worse, it's computationally expensive to interpolate between orientations. Chris Hecker summed it up pretty well: "It's possible to prove that no three-scalar param-

eterization of 3D orientation exists that doesn't suck, for some suitably mathematically rigorous definition of suck." I did say that I would try to keep the math to a minimum. While Euler angles are easy to understand, we need something that overcomes the weaknesses of using Euler angles for rotational animation. This is where something called a *quaternion* can save the day (see the sidebar: Pop Quiz Hot Shot).

A quaternion is an extension to complex numbers consisting of a vector and a scalar. There's no use trying to picture a quaternion because it exists in four-dimensional space. In the spirit of keeping the math to a minimum, let's review the key features of quaternions. A unit length quaternion is perfect for representing a rotational orientation of an object. Java3D supports a unit quaternion with the *Quat4f* class. As the name implies, it consists of four floating-point numbers to make up the vector and scalar components of the quaternion. It's straightforward to convert Euler angles to a quaternion as shown in Figure 7.

Performing successive rotations with quaternions is as easy as multiplying them together. When compared to the traditional rotational matrix approach, quaternion multiplication (the details of which we won't cover here) and orientation interpolation is much more efficient, making it ideal for animating our rotating particle shapes. To animate the rotation, we need to specify the angular velocity in the vector portion of a quaternion. The angular velocity quaternion used to calculate the time differential of a quaternion is shown in Figure 7. The time differential can be used to interpolate quaternions, which helps us spin objects. That was probably the world's shortest description of quaternions, so be sure to review the references if you need more detail. Let's put this new knowledge to work in our shape particle system.

When shape particles are about to be emitted, the orientation is assigned through the use of Euler angles. The angular velocity is also assigned using the now familiar central value and variance approach discussed above. The orientation and angular velocity is converted into quaternions by the particle. When the particle is updated, the quaternion differential is calculated using the time interval of the particle system manager as described in Figure 7. Finally, the new orientation quaternion is set on the *Transform3D* of the shape along with the new position and scale and Java3D rotates the shape.

```
Vector3f translation = new Vector3f();
translation.set(aParticle.getLocalPosition());
aTransform3D.set(aParticle.getOrientation(),
    translation,
    aParticle.getScale());
```

Quaternions are put to work in the *ColorCubeParticleSystemExample* example. This example uses the Java3D *ColorCube* utility class for the rotating particles and includes the *Gravity* and *BounceShape* influences. When running the example, note how the spinning is affected by the collision with the ground. There is a bit of physics at work in the *BounceShape* influence to calculate the torque

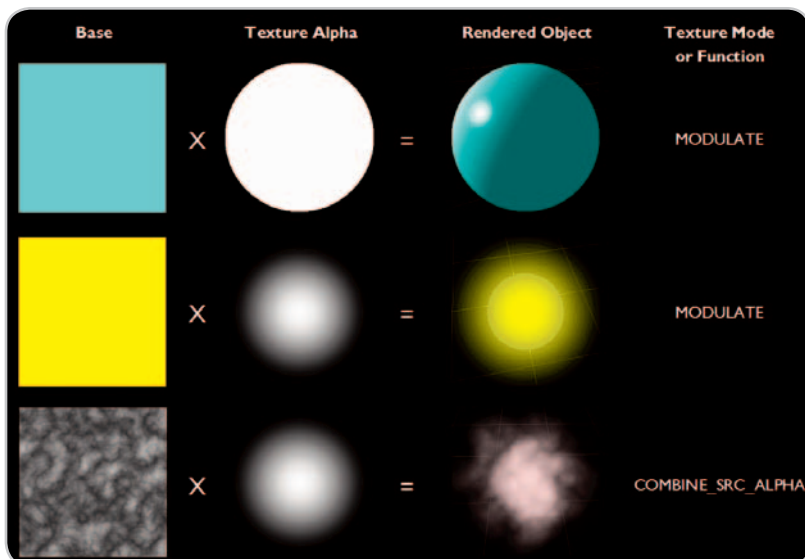


Figure 9 Ambient color or texture can be combined with material and texture transparencies to render a variety of implicit surfaces.

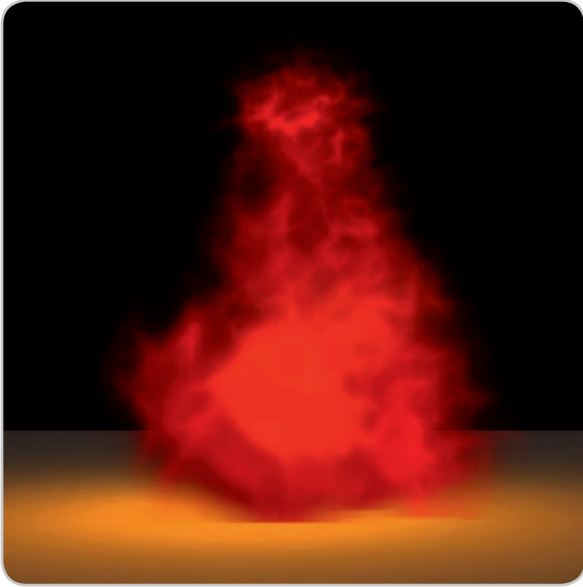


Figure 10 Real-time flames are possible with the particle system.

of the impact and the resulting spin. You can see that the influences provide an extension point to begin adding physical simulation aspects to the particle system. You can add your own Java3D shape as a particle by creating a factory and assigning it to the particle system as shown in Listing 2.

How about simulating some weather? Clouds as particles would be a powerful tool for steam, smoke, and even a tornado. However, you might be thinking, “But a cloud would take thousands of particles!” All it takes is one particle with just the right shape, an implicit shape to be precise.

Implicit Surfaces

Java3D excels at rendering objects with distinct geometries. How can we render difficult to describe electron density fields, flames or clouds? Complex organic or other vaguely defined objects can be rendered with an approach called *implicit surface modeling*. This approach is known by several names in the literature including blobby molecules, metaballs, and soft objects. We’ll use the term *implicit surface* because the shape of the object is implicitly determined by a function such as a sphere or ellipsoid. The implicit function is combined with other attributes to create the resulting object. For example, a spherical function can implicitly determine the shape of a cloud without making the cloud look like a ball. Vaguely defined objects like clouds cannot be defined with triangles, but with a visual trick called a billboard as shown in Figure 8.

A billboard is a geometrically flat image whose orientation is aligned with the view. You might know billboards better as sprites from older computer games. Games like Doom used eight different views for each monster, showing the appropriate image on the billboard based on the direction the monster was heading and the direction the player was facing. You never saw the edge of the image because the billboard was always ori-

ented along a vertical axis to face the player. Java3D includes support for billboards with the *OrientedShape3D* class.

OrientedShape3D goes beyond the Doom-style sprites to include any geometry with orientation about either an axis or a point. You might consider using an axis-oriented *OrientedShape3D* object if the user cannot move above the object and look down on it. Doing so would spoil the illusion since they would be able to see the image on edge. For this article, we’ll use point-oriented *OrientedShape3D* objects so that they face the user regardless of the view position. We will use a flat geometry, but through the use of textures, transparency, and vertex normals, we’ll be able to give the oriented shapes the feeling of depth.

Our overall strategy to implement implicit surfaces is to combine the material color and transparency with an alpha-enabled texture on an *OrientedShape3D* object as shown in Figure 9. The implicit surface is a sphere and our flat geometry slice of the sphere is a circle. The first texture is fully opaque within the sphere and fully transparent outside of the sphere. The ambient material color is combined with the *MODULATE* texture mode and vertex normals to create the fake sphere. While the geometry of the implicit surface is a flat *QuadArray*, the vertex normals are based on the implicit sphere. You can see from the rendered object that Java3D lights the flat geometry as though it is a real sphere. While this may not be a practical use of implicit surfaces, it does demonstrate the effectiveness of the approach. By varying the texture transparency based on the distance from the center of the sphere, we can create a fuzzy ball as shown in the second example. The final example uses the same varying transparency with Perlin noise-based turbulence texture to create a cloud.

The source code includes the *ImplicitSurface*, *FakeSphere*, *FuzzyBall* and *CloudPuff* classes to implement the examples in Figure 9. The *ImplicitSurface* class extends *OrientedShape3D* to support the others. The *CloudPuff* class can be used to implement volumetric fog without our particle system. Because these are all Java3D shapes, they can be used as particles in our shape particle system to create awesome special effects.

Implicit Surface Particles

Implicit surface particles depend heavily on transparency and rendering these objects takes additional care. Java3D logically renders our scene back to front based on the position of the view. This means that visible distant objects are rendered first and the closest objects are rendered last. This depth sorting ensures that the closer objects properly obscure the more distant objects. Ordinarily Java3D draws all of the opaque objects back to front, followed by the transparent objects. The transparent objects are not depth sorted and look incorrect when we use our transparent particles. Java3D depth sorts transparent objects if we call the following method on the view object:

```
setTransparencySortingPolicy(View.TRANSPARENCY_SORT_GEOMETRY)
```

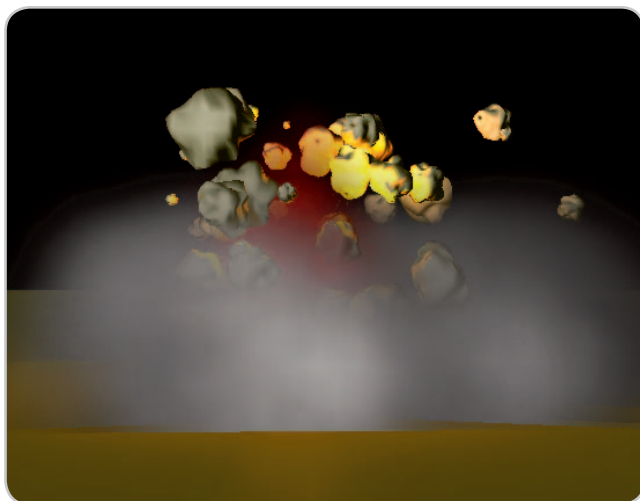


Figure 11 Implicit surfaces and shapes can be used together to create explosions.

Using this sorting policy correctly sorts transparent objects back to front. Now we can use our implicit surface particles to create more advanced examples.

Use Enough Dynamite There, Butch?

The source code included with this article includes several destructive examples worth mentioning. The tornado in Figure 5 was accomplished with the combination of three particle systems. The *TornadoExample* includes a particle system for the swirling clouds, the twisting funnel, and the resulting debris. Additional destruction is possible with fire as shown in Figure 10.

The *FlamesExample* combines the use of ridged fractal noise and light-emitting particles to create a real-time fire. The implicit surface particles are textured with a variety of Perlin noise to create the flame. The particle system shoots the particles straight up while shrinking the particles and making them more transparent as they age. Invisible *Phantom* particles are used to move and dynamically

attenuate point lights to make the fire appear to flicker. The *BlackHoleExample* sucks asteroids into a black abyss by using the *Attract* influence. And finally, everything we have done is demonstrated in the explosive finale shown in Figure 11.

The *ExplosionExample* combines most of the features of our particle system into one example. The example begins with a lit stick of dynamite next to large boulder. The fuse burns as a motion blurred point particle system. When the fuse is gone, the boulder explodes with dust, spinning rocks, fire, smoke and lights combined to complete the effect.

Acknowledgments

I would like to thank Ben Moxon and Brad Myers for reviewing this article. Star Trek is a registered trademark of Paramount Pictures. Doom is a registered trademark of id Software. ☺

References

- Ebert, D.S.; Kenton Musgrave, F.; Peachey, D.; Perlin, K.; and Worley, S. (2003). *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann Publishers.
- Bobick, N. "Rotating Objects with Quaternions." *Game Developer*. February 1998. Also available at <http://www.gamasutra.com/>.
- Hecker, C. "Physics – The Next Frontier." *Game Developer*. October/November 1996, December 1996/ January 1997, March 1997, and June 1997.
- Jacobs, M. "Casting Perlin's Movie Magic in Java3D." *JDJ*, Vol. 10, issue 3.
- Lander, J. "Better 3D: The Writing Is on the Wall." *Game Developer*. March 1998.
- Lander, J. "The Ocean Spray in Your Face." *Game Developer*. July 1998.
- Reeves, W.T., "Particle Systems – A Technique for Modeling a Class of Fuzzy Objects." *ACM Transactions on Graphics*, Vol. 2, No. 2. April 1983, Pages 91–108.

Listing 1

```
ParticleEmitter pe = new ParticleEmitter(
    new PointGenerationShape(Math.PI/4),
    400, // emission rate
    50, // emission rate variance
    8.5f, // velocity
    2, // velocity variance
    3, // lifetime
    1, // lifetime variance
    1500 // emitter lifetime
);
pe.addInfluence(new FadePoint());
particleSystem = new PointParticleSystem(pe, new Color3f(0,1,0));
// Adding the particle system to the scene not shown here
// Add a behavior to manage the particle system animation
ParticleSystemManager manager = ParticleSystemManager.getCurrent();
manager.setSchedulingBounds(bounds);
objRoot.addChild(manager);
```

Listing 2

```
float rate = (float) Math.PI/2;
Vector3f particleRotationRate = new Vector3f(rate, rate, rate);
```

```
Vector3f particleRotationRateVariance =
    new Vector3f(rate/2, rate/3, rate/4);

ParticleEmitter pe = new ParticleEmitter(
    new DiskGenerationShape((float)Math.PI/8, 2.5f,
    1.5f),
    4.3f, // emission rate
    1.2f, // emission rate variance
    22f, // velocity
    7f, // velocity variance
    particleRotationRate,
    particleRotationRateVariance,
    7.5f, // lifetime
    1.5f, // lifetime variance
    100 // emitter lifetime in seconds
);
pe.addInfluence(new Gravity());
pe.addInfluence(new BounceShape());

IShape3DFactory factory = new ColorCubeFactory();

Shape3DParticleSystem particleSystem =
    new Shape3DParticleSystem(pe, factory);
```



[Engage and Explore] the technologies, solutions and applications that are driving today's **Web services** initiatives and strategies...



Coming to a City Near You ▶

Web Services Edge Fall Conference Series

3 Dynamic Conference Programs Targeting Major Industry Markets

20+ seminars within 5 tracks will address the hottest topics & issues:

- ▶ Web Services: The Benefits and Challenges
- ▶ Web Services Security
- ▶ SOA (Service-Oriented Architecture) and ESB (Enterprise Service Bus) Strategies
- ▶ Interoperability, Incremental Integration, & Open Source
- ▶ The Management Process in Developing a Web Services Strategy

Why Attend:

- ▶ Improve the return on your technology investment
- ▶ Develop & sharpen your strategy and identify key action steps
- ▶ Find new ways to reach and impress customers with Web services
- ▶ Maximize the power of your enterprise
- ▶ Protect your business from security threats
- ▶ Assess Web services as a viable option

Program Features:

- ▶ Keynotes
- ▶ Tutorials
- ▶ Panel Discussions



Attention Exhibitors:

- ▶ An Exhibit-Forum will display leading Web services products, services, and solutions

Register Today! www.SYS-CON.com/Edge2005

Sponsored by

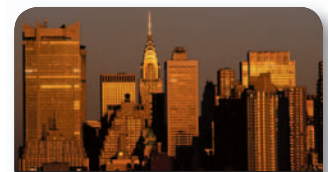


*Call for Papers email: grisha@sys-con.com



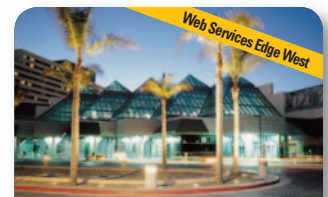
The Westin
Washington, D.C.
Washington, D.C.
September 7-8, 2005

1



Helmsley Hotel
New York, NY
September 19-20, 2005

2



The Westin Santa Clara
Convention Center
Santa Clara, CA
October 24-25, 2005

3

For Exhibit and Sponsorship Information ▶ **Call 201 802-3066**

Reviewed by
Michael Sayko

MKS Integrity Suite 2005

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult as establishing the detailed technical requirements...Therefore the most important function that software builders do for their clients is the iterative extraction and refinement of the product requirements.”

– Fred Brooks in *No Silver Bullet*
– *Essence and Accident in Software Engineering*

Experienced Java developers recognize that capturing and refining the requirements is one of the most important, yet difficult, parts of building a software system. An iterative software development process that incorporates the right tools can facilitate effective requirements management.

Product Background

MKS added requirements management to their software configuration management (SCM) product offerings with the recent release of Integrity Suite 2005. Although the enhancements to Integrity Suite encompass more than requirements management, this review focuses on the extensions to support requirements management since these extensions are the distinguishing features of the 2005 product.

Rather than developing a requirements management tool from scratch, or acquiring an existing product, MKS incorporated new features into their change management tool, Integrity Manager. These features allow Integrity Manager to collect and manage requirements. In addition, Integrity Manager can be used with Source Integrity to link requirements with source code files that are placed under version control.



Michael Sayko is a software configuration management consultant based in Austin, Texas. He is experienced with the practice of software configuration management from having served as a configuration manager on large, fast-paced software projects.

mss@acm.org

Product Architecture

MKS Integrity Suite 2005 is a J2EE application comprised of Integrity Server and Integrity Client.

Integrity Server manages the process items and source code files that reside on the server. It runs under an application server packaged and installed with the product. It uses FLEXlm as the license manager.

Integrity Client consists of three logical pieces: Source Integrity, Integrity Manager, and the Administration Client. Source Integrity is the version control interface, while Integrity Manager facilitates process and workflow management activities. Both Source Integrity and Integrity Manager share the same GUI. In a separate GUI, the Administration Client allows an administrator to perform common administrative tasks on the Integrity Server.

MKS

410 Albert Street
Waterloo, ON
N2L 3V3 Canada
Phone: 800 265-2797
Fax: 519 884-8861
Web: www.mks.com

Test Platform

Dell Inspiron 8100, 1.2GHz Intel Pentium III mobile, 512MB RAM, 40GB disk, Windows XP Professional Service Pack 2

Specifications Platforms

Server and Client: Windows NT/2000/XP, Unix, Linux

Pricing: Pricing for 10 users of the MKS Integrity Suite (including MKS Source Integrity Enterprise, MKS Integrity Manager, Integrity Server and MKS Requirements) is \$37,000

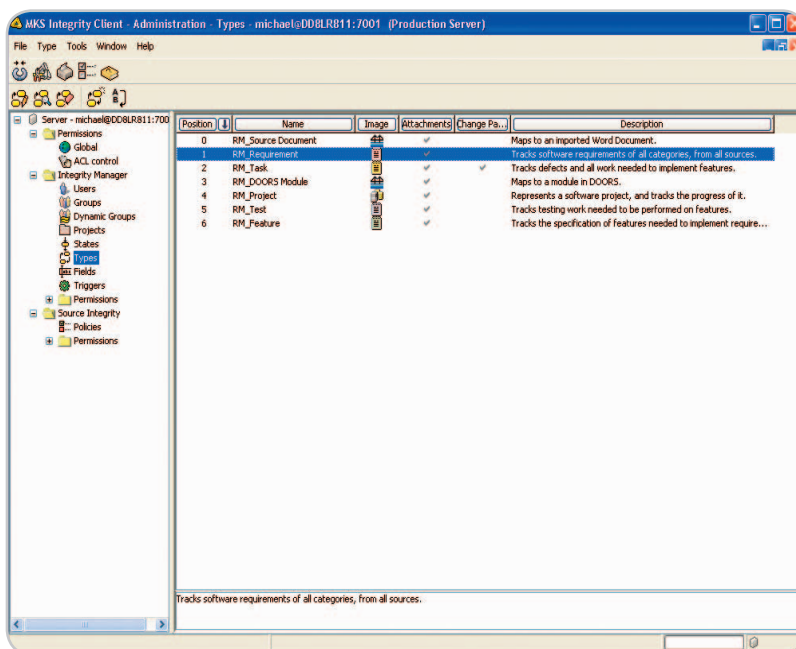


Figure 1 Requirements types in the Administration Client

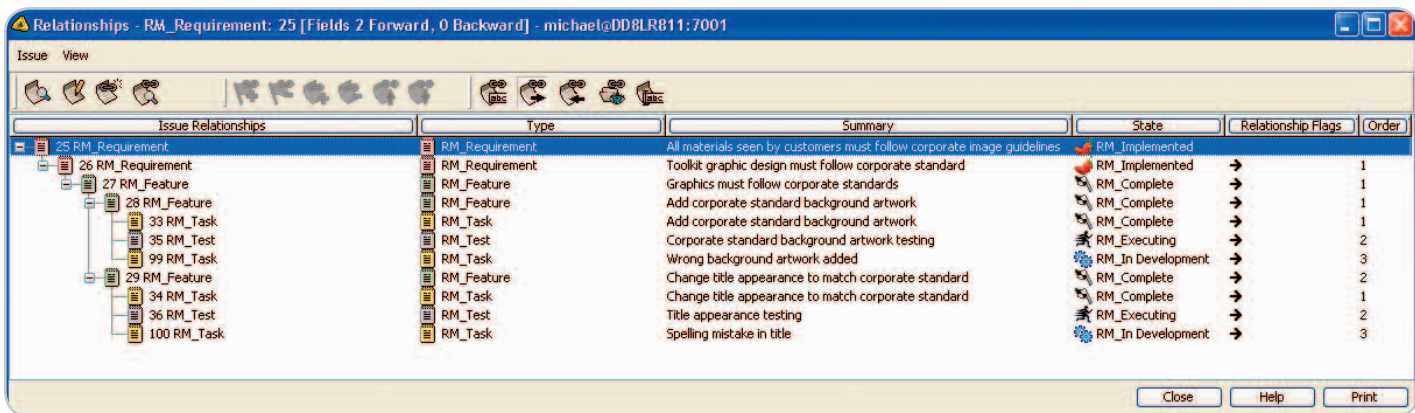


Figure 2 Relationship view

Installation

The Integrity Suite is distributed on two CDs. Installing the software is straightforward. For this evaluation, I installed both the Integrity Server and the Integrity Client on a 1.2GHz Wintel Notebook computer with 512MB RAM. I installed the Integrity Server using the embedded PointBase database. I configured the Integrity Server to use the flat file authentication scheme.

In addition to the embedded PointBase database, Integrity Server is designed to work with Oracle, MS SQL Server, and DB2 databases. Integrity Server stores process item data, including requirements, in the database. When using one of the supported commercial databases, files checked in to Source Integrity can also be stored in the database. During the installation of Integrity Server, the administrator needs to decide where version controlled files will be stored. If the database option is not selected, files placed under version control with Source Integrity are stored on a file system in Revision Control System (RCS) format.

MKS Requirements 2005

After installing the Integrity Suite, I proceeded to start the requirements management component. I soon realized that the requirements management component is really Integrity Manager. MKS added the following enhancements to Integrity Manager so that it could be used as both a requirements management repository and engine:

- Named issue relationships that link one issue to another
- A relationship view to visualize how issues are related

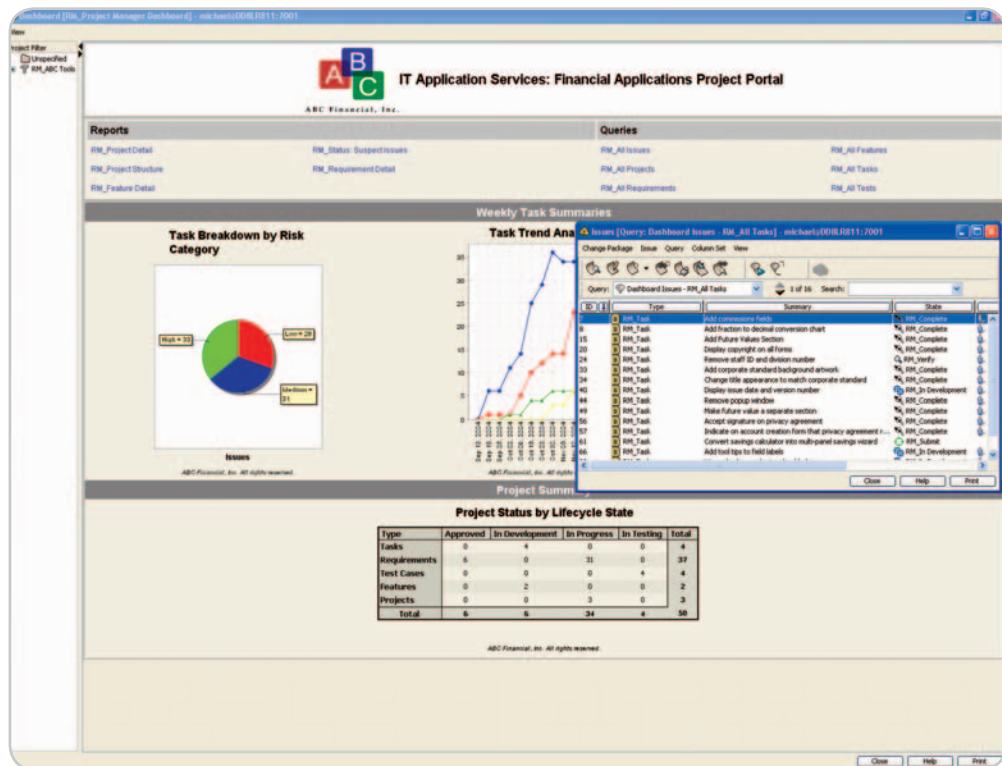


Figure 3 Project management dashboard

- Suspect links that control changes to requirements
- Integration with Microsoft Word to capture requirements defined in a Word document
- Integration with Telelogic DOORS to capture requirements defined in a DOORS module
- A process template consisting of five requirements-related processes

The *process template*, also called a *solution*, is named MKS Requirements 2005. While the process template is not required for requirements management, MKS created this template so their customers could visualize how to use the enhancements in Integrity

Manager to manage requirements. Although MKS describes the process template as “an illustration of MKS Integrity Suite’s requirements management capabilities,” I suspect that customers will want to use the template’s contents to model their requirements management process. Of course, customers can modify the process items defined by the template to address their specific needs.

Using Issues to Manage Requirements

To understand how requirements are managed by MKS Requirements 2005, you need to recognize that Integrity Manager uses *issues* to model process items. Each issue consists of *fields* that store data. An issue

transitions through a series of *states* to model a process workflow. Each issue is defined by its *type*. When a user creates issues from a type, the user can locate these issues with a *query*. A user-defined query simply selects and lists issues that meet certain criteria.

An administrator defines fields, states, and types using the Administration Client. A user then creates an issue (i.e., an instance of a type) and a query using Integrity Manager. Requirements, features, and tasks are examples of issues that can be managed by Integrity Manager. Prior to the release of Integrity Suite 2005, users of Integrity Manager created issues, such as change requests, to track software development activities like fixing a defect or adding an enhancement.

After enhancing Integrity Manager to support requirements management, MKS developed the Requirements 2005 process template to model requirements artifacts. The template consists of seven types (Project, Requirement, Source Document, DOORS Module, Feature, Task, and Test) that take advantage of the enhancements to Integrity Manager. Keep in mind that these types are administrator defined, rather than a base feature of Integrity Manager. For this reason, they can be used as is or modified through the Administration Client. Figure 1 shows how to access these types using the Administration Client.

Tracking Requirements, Analyzing the Impact of Changes, and Visualizing Project Status

The strength of MKS Requirements 2005 is the way in which it allows users to track requirements, analyze the impact of the inevitable changes to requirements, and visualize the status of a project.

When using the issues in the process template, a *requirement* is defined by *features* that are implemented as *tasks* and then validated through *tests*. Figure 2 depicts a chain of relationships from requirements to features to tasks. Although not shown in this figure, *change packages* link tasks to the source code files that implement them. This is the premise behind *task based development*. Each check-in to the version control system

is associated with a fine-grained development task. One benefit of task based development is that builds of the software application can be described by the features implemented, rather than just the source files modified. This makes the construction of a software application meaningful to a broader audience. Project managers, software testers, and end users can identify the features and fixes incorporated into every build of the software application. MKS Requirements 2005 extends task based development to *requirements based development* because it maintains relationships from requirements to features to tasks to the versions of source code files that implement the requirements. Now every build can also be described by the requirements that it implements.

Advocates of agile software development recognize that requirements evolve from their inception to their realization in working software. MKS Requirements 2005 supports requirements changes through *suspect links*. Suspect links are flags on relationship fields that are triggered when a requirement is changed. This allows dependent features, tasks, and tests to be marked as needing to be reviewed for the impact of a requirements change.

Finally, MKS Requirements 2005 allows team members to view the status of project using a *project management dashboard*. The dashboard is a real-time view of project data that provides

for interactive drill down to details. Figure 3 shows a project management dashboard with project status graphs and links to reports and queries.

Summary: Advantages of an Integrated Requirements Management Solution

The manner in which MKS integrated requirements management capabilities into Integrity Manager demonstrates the flexibility and extensibility of this process modeling and workflow management tool. Requirements, features, and tasks are managed like any other issue (i.e., process item) stored in the Integrity Manager database. One clear benefit of building requirements management artifacts from issues is that the artifacts can be linked to affected source code files using change packages. No integration effort is required to facilitate traceability of requirements and change management from the same repository.

While the requirements management components, workflows, and process rules described in the product documentation may appear to represent a rigid model, a closer look at the process template shows otherwise. The Integrity Manager documentation contains detailed, but easy to follow instructions for extending any Integrity Manager component, including the issues in the process template. By following these instructions, the components in the process template can be tailored, in a straightforward manner, using the Integrity Manager GUI. ❖

JDJ Product Snapshot

Target Audience: All members of the software development team including software configuration managers, developers, business analysts, and project managers.

Level: All levels, from beginner to expert.

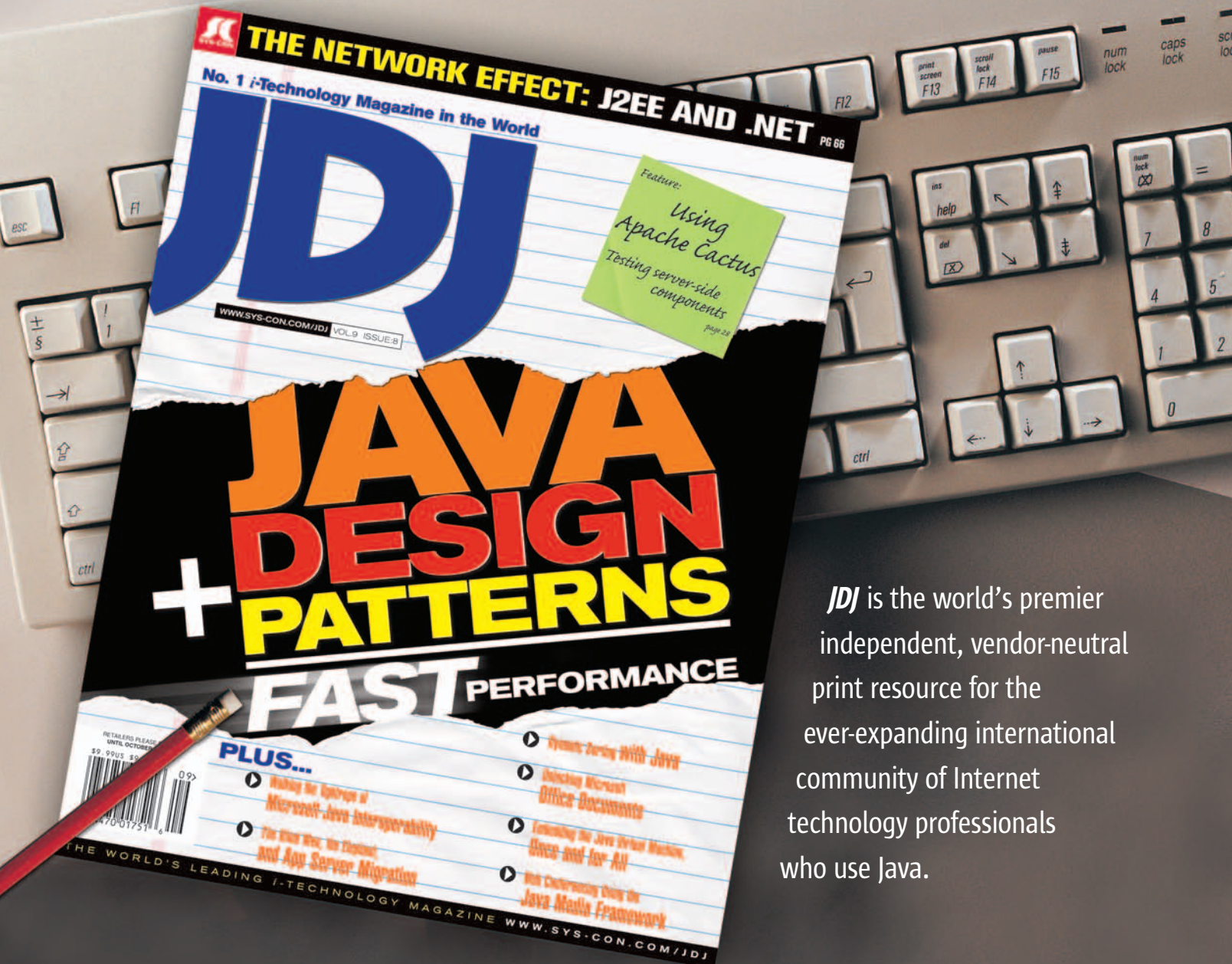
Pros:

- No integration required. MKS Requirements 2005 is a complete and self-contained requirements management solution because it's built with the process modeling and workflow management tool Integrity Manager. Contrast the approach followed by MKS to the model used by other tool vendors. Other requirements management products must be integrated with a software configuration management tool to provide traceability from requirements to source code.
- Facilitates requirements based development. Relationships coupled with change packages link requirements to the source code files that implement them. This also provides the traceability needed for audits, including software baseline audits.

Cons:

- First generation product. Even though MKS Requirements 2005 is built with the mature Integrity Manager, potential enhancements will be identified as MKS Requirements 2005 is used in production environments.

The World's Leading Java Resource Is Just a >Click< Away!



JDJ is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of Internet technology professionals who use Java.

Only **\$69⁹⁹**

ONE YEAR
12 ISSUES

Subscription Price Includes **FREE** JDJ Digital Edition!

www.SYS-CON.com/JDJ
or **1-888-303-5282**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**SYS-CON
MEDIA**
The World's Leading
i-Technology Publisher

Jtest 7.0

by Parasoft

Reviewed by Venkat

Java Technology completed its 10-year anniversary recently. Sun announced that there are over 4 million developers using the Java language, with thousands more joining their ranks. While a small percentage of developers can be called experts or skilled in the complete software development life cycle, a vast majority of them typically try to understand the requirements handed down to them and code to meet such requirements. Most projects (about 60%) go above budget and time, which places additional pressure on the developers trying to deliver an application. Coding standards, thorough unit testing, best coding practices – all take a back seat to the primary goal of delivering some code that meets functional requirements. Software architects have long tried to enforce a uniform coding and testing practice on their teams. Now, with the release of Jtest 7.0 from Parasoft, it's possible to deliver code that follows best practices and is well unit tested.



Product Description

Jtest is an automated Java code analysis and unit test generation product. It comes with over 500 coding standard rules built-in and also provides a mechanism to correct over 200 of these violations automatically. Rules can be customized without coding, and user-defined rules can also be created. It automatically generates and executes JUnit tests and allows users to extend these tests. Jtest has been standardized on the now ubiquitous Eclipse platform and is available as a stand-alone and as a plug-in to Eclipse. It's available on

Windows, Linux, and Solaris platforms. Jtest makes static code analysis a breeze and it should be easy for even beginners to analyze their code for compliance to standards as well as to generate unit tests.

Installation

I downloaded and installed the stand-alone version of the software, which is available for download from the Parasoft Web site after going through a simple registration process. A license key is e-mailed to the provided address. Installation of the software is straightforward and I just went with the provided default values. I had JDK 1.5 already available on my machine. Jtest stand-alone installs the entire Eclipse runtime and associated files with it. Once you install it, the first time you run the product you can go to the preferences under the Windows menu and put in your license key information. You're now ready to go.

Jtest Usage and Benefits

If you are familiar with Eclipse, there is practically no learning curve. You can switch perspectives, create a Jtest project from existing source folders, and be on your way. There are also several example projects available for newbies to learn how to use Jtest. The Java, CVS, and debug perspectives are all available as a consequence of Jtest built on the Eclipse platform. You can create Jtest configurations that can then be applied to a project to conduct static code analysis. A subset of the available 500 rules can be enabled and custom rules can also be added

Parasoft

101 E. Huntington Drive
Monrovia, CA 91016

Web: www.parasoft.com

Phone: (888) 305-0041

E-mail: info@parasoft.com

Test Platform

JDK 1.4.2 and 1.5, HP Compaq nx9600, Windows XP Professional with Service Pack 2, Pentium 4 @ 3.6 GHz, 1GB RAM, 80GB hard disk

Specifications

Platforms: Windows, Linux, and Solaris

Pricing: Pricing starts at \$3,495 for a single user, machine-locked license.

Support: Online technical support; dedicated technical support with a 1-800 number, 9:00 a.m. to 5:00 p.m., Monday through Friday; overall Parasoft solution including testing process, testing analysis, and professional services support.

to a configuration. Figure 1 shows the stand-alone version of Jtest 7.0 with available examples.

I perceived several benefits to using Jtest. It prevents errors from entering into your code in the first place. For new software development projects, code development can be treated in a strict manner and the coding process controlled to follow established coding standards. Applying coding standards is easy with the over 500 built-in coding standard rules that enforces code design and construction.

Issues associated with resources like JDBC connections not being closed properly are eliminated upfront. This ensures that denial of service (internal and external denial

“With the release of Jtest 7.0 from Parasoft, it's possible to deliver code that follows best practices and is well unit tested”

Subscribe Today!

— INCLUDES —
FREE DIGITAL EDITION!
(WITH PAID SUBSCRIPTION)
GET YOUR ACCESS CODE INSTANTLY!



The major infosecurity issues of the day... identity theft, cyber-terrorism, encryption, perimeter defense, and more come to the forefront in ISSJ the storage and security magazine targeted at IT professionals, managers, and decision makers

Editorial Mission

Greater Collaboration and Efficiency Through Education

- ✓ ISSJ's editorial mission is to showcase proven solutions that will guide, motivate, and inspire senior IT and business management leaders in the planning, development, deployment, and management of successful enterprise-wide security and storage solutions.
- ✓ ISSJ brings together all key elements of data storage and protection, and presents compelling insight into the benefits, efficiencies, and effectiveness gained by focusing on these two critical areas of IT simultaneously.
- ✓ ISSJ is an objective, critical source of information that helps storage and security managers make informed management decisions about what is working today, and what they need to plan for tomorrow, and is the only publication that focuses exclusively on the needs of IT professionals who are driving the enterprise storage architecture/infrastructure while pursuing and incorporating the latest security technologies.
- ✓ ISSJ achieves our mission by delivering in-depth features, practical "how-to" solutions, authoritative commentary, hard-hitting product reviews, comprehensive real-world case studies, and successful business models, written by and for professional IT storage and security practitioners.

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only \$39⁹⁹

ONE YEAR
12 ISSUES

www.ISSJournal.com

or

1-888-303-5282

SYS-CON
MEDIA

The World's Leading IT-Technology Publisher

Snapshot

Target Audience: Java developers, software architects

Level: Beginner to advanced

Pros:

- Easy to use for all user levels
- Static code analysis is quite extensive
- Unit tests generated are of an excellent standard
- Integrate existing JUnit tests into Jtest

Cons:

- Integration only with Eclipse and not NetBeans
- Limited support for Java 5.0 (JDK 1.5)
- Sniffer functionality supported for JDK 1.4.x and JDK 1.5 JVMs

attacks that leverage the exposure of object resources are eliminated. For example, it's possible for a poorly written application to access your applications and induce it to throw an exception. This can leave a connection open if you don't close your connection in a finally clause. This scenario can be easily caught by one of the built-in rules, and good coding practices can be enforced.

The unit testing features of Jtest are exciting and are based on automa-

tion. Unit tests designed to break your code are generated automatically. The generated unit tests can run in batch mode overnight on the server.

The prebuilt Jtest configuration has several rules such as avoiding empty try/catch blocks, handling exceptions and errors, and assignment within condition blocks. There is the notion of an object repository where you can create complex structures. Jtest can read this repository when it's generating tests. You can also extend generated unit tests, automatically insert assertions, as well as get reports on the code coverage and metrics.

The team configuration manager (TCM) allows you to enable sharing of a configuration across a team. You can upload test results to TCM. When a developer starts his or her IDE in the morning he or she can see the nightly batch test reports that were run on the server.

Last but not the least, the most exciting of all available features in Jtest 7.0 is test case sniffing. This is the ability of Jtest to monitor JVM execution and based on that create functional tests. Currently, only JDK

1.4.x is supported. The API to hook into the JVM has undergone a revision from JVMPI to JVMTI and that could possibly explain why Parasoft may take some time to catch up on sniffing the JVM under JDK 1.5.

Jtest can send data from various tests to the Parasoft Group Reporting System (GRS). Based on the data sent to GRS, the configuration of Jtest on developer machines can be modified. Project Management can use GRS reports to allow them visibility on overall project progress. GRS provides trending over time, graphics, etc.

Summary

Overall, Jtest 7.0 is definitely an able ally in the software developer camp. It can save a lot of effort in unit testing as well as allow a team to enforce and follow uniform coding standards. For small projects, the price can probably be a little steep. If you have a large project with multiple developers, I would definitely encourage using a code analysis tool and also complement your unit testing efforts. Parasoft's Jtest definitely fits the bill in that regard. ☺

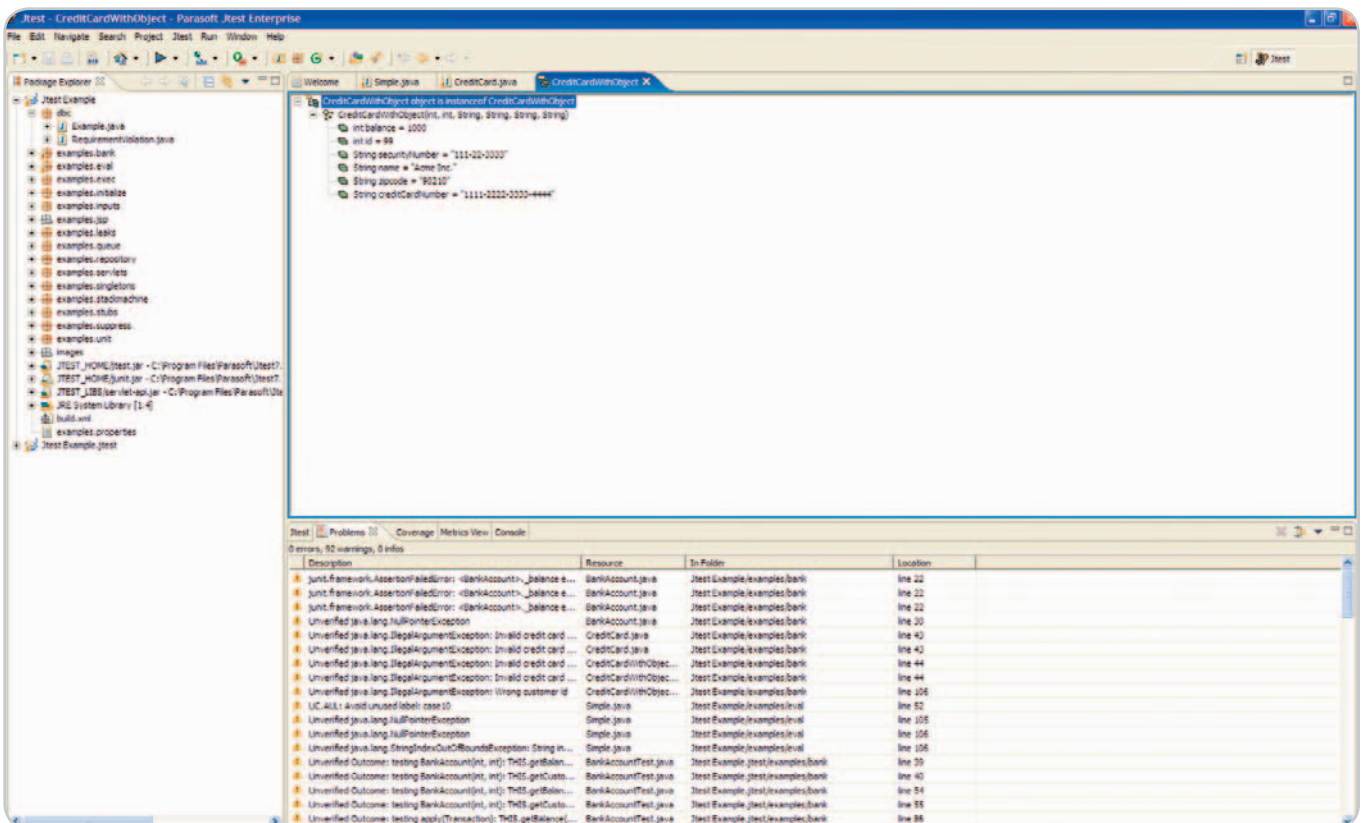


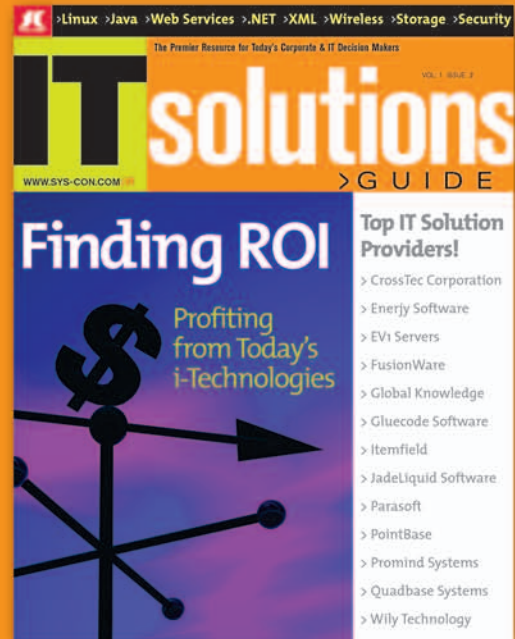
Figure 1 Stand-alone version of Parasoft Jtest 7.0

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	4
Borland	www.borland.com/jbuilder	831-431-1000	7
ceTe Software	www.dynamicpdf.com	800-631-5006	49
ClearNova	www.clearnova.com	877-223-8651	35
Common Controls	www.common-controls.com	+49 (0) 6151/13 6 31-0	59
DataDirect	www.datadirect.com/jdj	800-876-3101	Cover IV
EV1 Servers	www.ev1servers.net	800-504-SURF	47
Google	www.google.com/jdj	650-253-0000	41
InetSoft	www.inetsoft.com/jdj	888-216-2353	29
Information Storage & Security Journal	www.issjournal.com	888-303-5282	69
InterSystems	www.intersystems.com/free8p	617-621-0600	13
IT Solutions Guide	www.sys-con.com/it	888-303-5282	71
JadeLiquid Software	www.webrenderer.com	+61 3 6226 6274	31
Java Developer's Journal	www.sys-con.com/jdj	888-303-5282	67
JavaOne Conference	www.java.sun.com/javaone/sf	866-382-7151	57
Jinfonet	www.jinfonet.com/jp6	301-838-5560	17
M7	www.m7.com/power	866-770-9770	27
Microsoft Visual Studio	www.msdn.microsoft.com/visual		Cover II, 15
NCL	www.nclt.com/jdj	+353 1 6761144	45
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	55
ODTUG Conference	www.odtug.com/2005_conference_location.htm	888-627-7033	51
Parasoft Corporation	www.parasoft.com/jtest	888-305-0041	11
Perforce	www.perforce.com	510-864-7400	9
Prolifics	www.prolifics.com	800-675-5419	53
ReportingEngines	www.reportingengines.com	888-884-8665	21
SleepyCat Software	www.sleepycat.com/bdbje	510-597-2128	37
Software FX	www.softwarefx.com	800-392-4278	Cover III
Synaptris	www.intelliview.com/jdj	866-991VIEW	25
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	73
Visual Paradigm	www.visual-paradigm.com	408-426-8212	33
WebAppCabaret	www.webappcabaret.com/jdj.jsp	866-256-7973	39
Web Services Edge Fall 2005	www.sys-con.com/edge2005	201-802-3066	63
Xythos	www.xythos.com	888-4XYTHOS	43

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.

Reach Over 100,000 Enterprise Development Managers & Decision Makers with...



Offering leading software, services, and hardware vendors an opportunity to speak to over 100,000 purchasing decision makers about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management

Don't Miss Your Opportunity to Be a Part of the Next Issue!

Get Listed as a Top 20* Solutions Provider

For Advertising Details
Call 201 802-3021 Today!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE.

OSS: The Market Landscape



Philippe Lalande

Why the OSS industry can – for the first time ever – produce open, implementable, and certifiable standards

This past month the JCP Executive Committees met in Nice, France, in concert with the TeleManagement World conference, where the worldwide telecommunications industry gathered to address technical, operating, and business issues related to the back office systems that they use to run their business with their networks and services portfolios. These back office systems are called Operation Support Systems and Business Support Systems (OSS/BSS), or OSS for short.

My guest this month is Philippe Lalande, head of the OSS through Java Initiative at Sun. Philippe created and drove the Java Specification Request (JSR3) under the JCP, and founded the OSS through Java Initiative, which he continues to lead today. You'll find it interesting to learn directly from the guy at the heart of it all how the level of complexity of OSS systems is driven down by an order of magnitude by Java technology, the JCP, and Java.net, all of which have been instrumental in re-shaping an entire vertical industry segment, and in driving up these systems' level of agility.

— Onno Kluyt, chair,
Java Community Process (JCP)

The OSS/BSS market is worth roughly \$50 billion on its own, but directly impacts half a trillion dollars in operating costs. Owning an OSS is actually what characterizes a service provider. Traditional wireline operators own their networks, while MVNOs or mobile virtual network operators run their services on someone else's networks. But all service providers own their OSS. It's their tool to do business.

Successfully running a communications service provider business today requires simultaneously operating

large network infrastructures and portfolio of services to cut operating costs, bring new services to market faster and drive customer satisfaction. This exercise is heavily dependent on software. Myriad applications are deployed across very large scale IT infrastructures to cover functional domains such as network management, service provisioning, fault correlation, inventory management, service quality monitoring, order entry, billing, or trouble ticketing. Business process automation requires integrating all these systems together, making sure they share some common information models across the OSS. Taken individually these applications aren't really exotic, and integrating any two of them isn't rocket science. But the devil's in the numbers.

On average, a tier-one service provider runs 1,500 different applications. Bell Canada runs 162 different billing systems, and BT recently inventoried 3,000 OSS applications across the company. Some of them were developed in-house, others were customized from commercial off-the-shelf products, and still others were inherited from mergers and acquisitions. Some of them are very modern, recently designed on the latest software technologies; others are over 20-years-old. All of them have to run 24/7/365 because every minute the network isn't costs several millions dollars. All of them have to deal in real-time with thousands of nodes, millions of customers, zillions of commercial transactions (such as a single phone call). If you've got the picture, welcome to the OSS integration nightmare!

This nightmare becomes hell with the increased complexity and competitive pressure that come with smart devices, with the convergence of voice,

data, entertainment, and soon utility computing. To support IP-based next-generation networks, industry de-regulation and other major telco trends, service providers around the world are embarking on multi-year endeavors like Sprint's so-called "transformation journey," British Telecom's "21st century network," France Telecom's "urbanization," and BellSouth's "competitive survival initiative." The ultimate goal of these strategic programs is to rationalize the IT infrastructures, evolve to more flexible component-based OSS systems, harmonize security, user interfaces, and other non functional features, and reduce the market fragmentation that results from decades of operating network silos and services hard-coded into the network elements, building bespoke telco-specific IT platforms, and trying to define standards that remained paper specifications but never made it to real implementation and adoption.

The OSS/J Approach

The OSS/J Initiative was originally founded by Cisco, Ericsson, Motorola, NEC, Nokia, Nortel, Telcordia, and Sun to fill the gaps in the OSS standardization landscape and foster a market of re-usable OSS components. To address interoperability issues, it was decided to focus on technology specific implementation, on mainstream enterprise technologies, and on making the life of developers easier and more exciting.

Rather than attempting once again to create the next-best-telco-specific OSS integration middleware (there are already approximately 400 of those proprietary things polluting the industry), OSS/J chose J2EE to provide the underlying middleware and to share the risk and the investment with other industries.

Philippe Lalande co-founded and leads the OSS through Java Initiative, and is on the board of the TMF. At Sun, he also created Java Management eXtensions technology (JMX) and is a recognized expert in applying Java and Web Services technology to create business value in the areas of management and telecommunications. In 2002 he received Sun's Chairman award for innovation. Prior to joining Sun, Philippe held technical and business responsibilities at Alcatel.

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO \$340 AND RECEIVE UP TO 3 FREE CDs!*



RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

3-Pack
Pick any 3 of our magazines and save up to **\$210⁰⁰**
Pay only \$99 for a 1 year subscription plus a FREE CD

- 2 Year – \$179.00
- Canada/Mexico – \$189.00
- International – \$199.00

6-Pack
Pick any 6 of our magazines and save up to **\$340⁰⁰**
Pay only \$199 for a 1 year subscription plus 2 FREE CDs

- 2 Year – \$379.00
- Canada/Mexico – \$399.00
- International – \$449.00

9-Pack
Pick 9 of our magazines and save up to **\$270⁰⁰**
Pay only \$399 for a 1 year subscription plus 3 FREE CDs

- 2 Year – \$699.00
- Canada/Mexico – \$749.00
- International – \$849.00

CALL TODAY! 888-303-5282

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

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

TO ORDER • Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

LinuxWorld Magazine

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 / Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 / Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 / Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 / Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 / Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

JDJ

U.S. - Two Years (24) Cover: \$144	You Pay: \$99.99 / Save: \$45 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$69.99 / Save: \$12
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 / Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$89.99 / Save: \$40
Intl - Two Years (24) \$216	You Pay: \$176 / Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

Web Services Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 / Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 / Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 / Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 / Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 / Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

.NET Developer's Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 / Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 / Save: \$14
Can/Mex - Two Years (24) \$168	You Pay: \$129 / Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 / Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 / Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

Information Storage + Security Journal

U.S. - Two Years (24) Cover: \$143	You Pay: \$49.99 / Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 / Save: \$39
Can/Mex - Two Years (24) \$168	You Pay: \$79.99 / Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 / Save: \$34
Intl - Two Years (24) \$216	You Pay: \$89.99 / Save: \$126 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$59.99 / Save: \$48

Wireless Business & Technology

U.S. - Two Years (12) Cover: \$120	You Pay: \$49.00 / Save: \$71 + FREE \$198 CD
U.S. - One Year (6) Cover: \$60	You Pay: \$29.99 / Save: \$30
Can/Mex - Two Years (12) \$120	You Pay: \$69.99 / Save: \$51 + FREE \$198 CD
Can/Mex - One Year (6) \$60	You Pay: \$49.99 / Save: \$10
Intl - Two Years (12) \$120	You Pay: \$99.99 / Save: \$20 + FREE \$198 CD
Intl - One Year (6) \$72	You Pay: \$69.99 / Save: \$2

MX Developer's Journal

U.S. - Two Years (24) Cover: \$143	You Pay: \$49.99 / Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 / Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$79.99 / Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 / Save: \$34
Intl - Two Years (24) \$216	You Pay: \$89.99 / Save: \$126 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$59.99 / Save: \$48

ColdFusion Developer's Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 / Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 / Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 / Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 / Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 / Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 / Save: \$2

WebSphere Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129.00 / Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 / Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 / Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 / Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189.00 / Save: \$75
Intl - One Year (12) \$132	You Pay: \$129.99 / Save: \$2

PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 / Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 / Save: \$31
Can/Mex - Two Years (24) \$240	You Pay: \$179.99 / Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 / Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 / Save: \$1

WLDJ

U.S. - Four Years (12) Cover: \$240	You Pay: \$99.99 / Save: \$140 + FREE \$198 CD
U.S. - Two Year (12) Cover: \$120	You Pay: \$49.99 / Save: \$70
Can/Mex - Four Years (24) \$240	You Pay: \$99.99 / Save: \$140 + FREE \$198 CD
Can/Mex - Two Year (12) \$120	You Pay: \$69.99 / Save: \$50
Intl - Four Years (24) \$240	You Pay: \$120 / Save: \$120 + FREE \$198 CD
Intl - Two Year (12) \$120	You Pay: \$79.99 / Save: \$40

*WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

Being relieved by J2EE of most of the typical framework issues, it was then possible for the initiative to focus exclusively on OSS functional APIs that implement existing telco OSS paper standards whenever possible and run on any compliant J2EE application servers. OSS/J has established pull-push relationships with a number of standards bodies, *TeleManagement Forum* being the cornerstone of that strategy. Rather than develop its own standards or variants, OSS/J had some of its members contribute resources to accelerate completion of some technology-neutral standards like TMF NGOSS and align with them.

Then, because the APIs address integration tax, interoperability, and other non-differentiating common pain points, we decided that each API should come with free-of-charge, near-production-ready implementation examples, as well as free-of-charge conformance test suites. Therefore, the next critical OSS/J decision was to adopt the JCP as its collaboration and process framework. OSS/J actually decided to enforce more demanding requirements above and beyond the JCP legal framework and created its own open collaboration and governance model. This enabled the initiative to impose active participation and measurable contributions from its members, create a consistent family of APIs, factorize design patterns across all APIs, and control the overall roadmap. It particularly wanted to avoid the useless proliferation of APIs that would be too specific for certain functions or network technologies.

The APIs are verified with real products, packaged, distributed, and licensed under a harmonized licensing and certification model that avoids possible differences from one API and spec lead to another. It also enables the initiative to substitute specification and maintenance leads when needed.

The initiative operates as a virtual company, with its own steering and decision processes and its own budget, although it's not a legal entity. The governance model is open and public. Interested readers can take a look at the *master agreement and the operating guide*.

Thanks to all these choices, OSS/J delivers open standards that are absolutely unique in the telecommunications industry:

- No other standard delivers publicly available, free-of-charge, testable, and certifiable implementations
- No other implementation solution is an open standard

What We Learned?

Recent OSS industry analyst reports like the one from Dittberner claim the middleware game is over. J2EE and its OSS/J plug-in are heading towards 80% market share by 2008. The JCP has absolutely delivered on its promises enabling the OSS industry for the first time ever to produce open, implementable, and certifiable standards. This process framework, extended with OSS/J governance model and associated to the J2EE design patterns practices, has helped us scope and design the OSS/J technology into a very small set of only 15 APIs covering a very large portion of OSS's functional space, and at the same time a broad spectrum of network technologies including wireline, wireless, broadband, and cable. The technology is now deployed in production across the planet, and the early adopters have achieved tangible benefits; DSL provider Covad recently said that it \completed an OSS integration project in four months that would have typically required two years – and their second project was done in two months.

So, can we claim victory today?

No. The network operators that have been voicing their frustration over incompatible interfaces for so long are paradoxically moving carefully and slowly. They were snake-bitten by the promises of previous standards like Q3 and previous technologies like CORBA that were too complex to implement and too specific to the OSS niche. The current business model with its 80% tax for non-re-usable customization and integration is the comfort zone of several vendors and buyers as well. They believe that by maintaining the status quo, they will protect their short-term cash cow or their personal piece of political power.

But long-term, the industry simply can't afford to keep wasting 80% of its investments. As the Yankee Group says, adopting OSS/J to build the next-generation OSS is a matter of survival.

So we still have some work to do, and there are still many opportunities for Java developers to surf the OSS/J wave.

What's Next?

The OSS/J technology is now stable. The initiative will complete the APIs now in progress through the JCP, and for each API will finalize the three integration profiles needed to cover the typical integration scenarios needed in the industry (JVT/EJB, XML/JMS, and XML/WS).

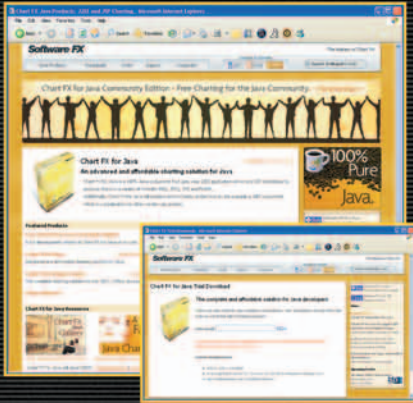
The early adopters have confirmed that the family of APIs is useful and solid, and serve the purpose they were designed for. The bet they made as the first OSS/J adopters has been rewarded by immediate and tangible benefits but this is not enough to get the technology rapidly embraced by the more conservative masses. The people who don't have the time, resources, or skills to dive into the standards and the technology need an ecosystem of certified products, adapters, extensions, modeling, development, and presentation tools plus qualified training and consulting services. The good news is that the same rationale that led OSS/J to choose J2EE at the beginning now leads to building that ecosystem on top of mainstream enterprise building blocks such as portal servers, directory servers, identity servers. And we believe this will bring another order of magnitude of simplification and cost savings to the industry.

We have recently created a number of *OSS/J-related open source projects on Java.net* to ignite this ecosystem. If it's successful, many will consider the game over, but I'm personally convinced that the most exciting part is still to come with the convergence of telco and the enterprise. The day utility computing services are sold like today's communications services, the most successful companies will run the most efficient and flexible OSS systems. With OSS/J and other Java management technologies such as JMX, Java developers have a baseline from which to play a major role in that evolution. We've created a placeholder for *service and business management projects* on Java.net, and I'm soliciting your ideas and energy to help turn this into a vibrant community. If we can connect the dots between telco OSS and enterprise service and business management the opportunities are unlimited. ☺

SoftwareFX

Zero To Chart

In Less Than An Hour!



9:04 am

To learn more about the Chart FX products visit www.softwarefx.com and decide which one is right for your platform and target. Then download the 30-day trial version or the Chart FX for Java Community Edition which is a FREE, non-expiring, full development version.

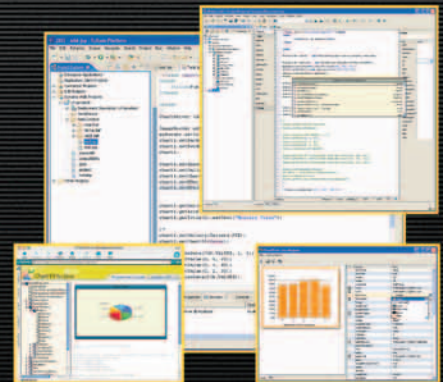
9:07 am

After download, simply install the product appropriate for your needs, platform and target environment.



9:13 am

Open the Chart FX for Java Designer to get started with the look and feel of your chart. Use your favorite IDE to add functionality and to populate the chart by connecting to your data source. The Chart FX Resource Center is also available to help with any questions you may have. Then deploy to your specific application server. ➤



9:58 am

Your application is then displayed with an active chart or image that makes any data look great!



Ready... Set... Download!



US: (800) 392-4278 • UK: +44 (0) 8700 272 200 • Check our website for a Reseller near you!

www.softwarefx.com

Remember the good old days?



Can you dig it? We miss shakin' our groove things, too. You can still get down to Nectarines and Spice's sweet and soulful sounds, but you wouldn't use an 8-track tape player to do it.

It's like that with Microsoft® SQL Server. Terrific database, but the JDBC driver available from Microsoft is old technology you wouldn't use today. You need better performance, reliability, and functionality. **DataDirect presents today's JDBC** – the SPECjAppServer/ECperf leader. Featuring Windows Authentication support, J2EE certification, and advanced 3.0 specification compliance for SQL Server 2000 and SQL Server 7.

Some things are better left in the past. Like that pale green leisure suit.

Why compromise on yesterday's technology?
Get current with **DataDirect Connect™ for JDBC**.

www.datadirect.com/jdj
(800) 876-3101


DataDirect™
TECHNOLOGIES