

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

JDJ

DECEMBER 2004 VOLUME:9 ISSUE:12

PERSONALIZE YOUR WEB APPLICATIONS

Reusable components can eliminate personalization code from your applications

PLUS...

- ▶ Understanding Portals and Portlets
- ▶ Moving to a Cluster...
- ▶ Providing a Complete Data Services Layer
- ▶ Using JAXB in J2EE-Based Enterprise Applications
- ▶ Using JNDI...
- ▶ Developer Testing Is 'In'

RETAILERS PLEASE DISPLAY
UNTIL FEBRUARY 28, 2005

\$9.99US \$9.99CAN

0 1 >



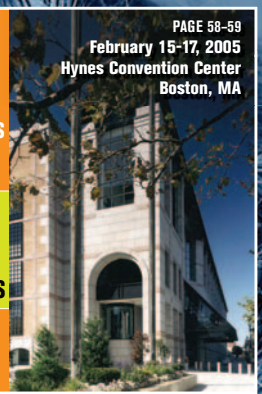
web services **EDGE**
conference & expo

PAGE 58-59
February 15-17, 2005
Hynes Convention Center
Boston, MA

- Join Over **3,000** Developers
- App Server Shoot-Out!

- **FREE** J2EE Tutorial
- JDJ Partner Pavilion
- Seminars and Case Studies

\$1,000,000
Software Giveaway



Your potential. Our passion.[™]
Microsoft

Feel free to think big.

With Visual Studio[®].NET 2003, you write less code than with Microsoft[®] Visual Basic[®] 6.0, so you can turn that big idea into reality faster than you ever thought possible. You get a new IDE that leverages your existing skills in a tool that gives you more robust code, improved IntelliSense[®], and better deployment. 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



Microsoft
Visual Studio

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

The *i*-Technology Right Stuff



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 Beatty-Lima**
 Executive Editor: **Nancy Valentine**
 Associate Editors: **Jamie Matusow**
Gail Schultz
 Assistant Editor: **Natalie Charters**
 Online Editor: **Martin Wezdecki**
 Research Editor: **Bahadir Karuv, PhD**

Writers in This Issue

Calvin Austin, Bill Burke, Keith Donald, Bill Dudney, Jeremy Geelan, Onno Kluyt, Tilak Mitra, David Purcell, Ken Ramirez, Thomas Smits, Rost Vashevnik, Daniel Vlad, Coach K. Wei, Joe Winchester

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

Subscriptions

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

888 303-5282
 201 802-3012

subscribe@sys-con.com

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

Editorial Offices

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

Java Developer's Journal (ISSN1087-6944) is published monthly (12 times a year) for \$69.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. Postmaster: Send address changes to: Java Developer's Journal, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

©Copyright

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

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

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

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

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.



Our search for the Twenty Top Software People in the World is nearing completion. In the *JDJ* tradition of empowering readers, we will leave the final “cut” to you, but here – in the meantime – are the top 40 nominations. More details on each nominee can be viewed online at www.sys-con.com/java.

In alphabetical order the nominees are:

- **Tim Berners-Lee:** “Father of the World Wide Web” and expectant father of the Semantic Web
- **Joshua Bloch:** Formerly at Sun, where he helped architect Java’s core platform; now at Google
- **Grady Booch:** One of the original developers of the Unified Modeling Language
- **Adam Bosworth:** Famous for Quattro Pro, Microsoft Access, and IE4; then BEA, now Google
- **Don Box:** Coauthor of SOAP
- **Stewart Brand:** Cofounder in 1985 of the WELL bulletin board
- **Tim Bray:** One of the prime movers of XML, now with Sun
- **Dan Bricklin:** Cocreator of VisiCalc, the first PC spreadsheet
- **Larry Brilliant:** Cofounder in 1985 of the WELL bulletin board
- **Sergey Brin:** Son-of-college-math-professor turned cofounder of Google, Inc.
- **Dave Cutler:** The brains behind VMS; hired away by Microsoft for Windows NT
- **Don Ferguson:** Inventor of the J2EE application server at IBM
- **Roy T. Fielding:** Primary architect of HTTP 1.1 and a founder of the Apache Web server
- **Bob Frankston:** Cocreator of VisiCalc, the first PC spreadsheet
- **Jon Gay:** The “Father of Flash”
- **James Gosling:** “Father of Java” (though not its sole parent)
- **Anders Hejlsberg:** Genius behind the Turbo Pascal compiler, subsequently “Father of C#”
- **Daniel W. Hillis:** VP of R&D at the Walt Disney Company; cofounder, Think Machines
- **Miguel de Icaza:** Now with Novell, cofounder of Ximian
- **Martin Fowler:** Famous for work on refactoring, XP, and UML

- **Bill Joy:** Cofounder and former chief scientist of Sun; main author of Berkeley Unix
- **Mitch Kapor:** Designer of Lotus 1-2-3, founder of Lotus Development Corporation
- **Brian Kernighan:** One of the creators of the AWK and AMPL languages
- **Mitchell Kertzman:** Former programmer, founder, and CEO of Powersoft (later Sybase)
- **Klaus Knopper:** Prime mover of Knoppix, a Linux distro that runs directly from a CD
- **Craig McClannahan:** Of Tomcat, Struts, and JSF fame
- **Nathan Myhrvold:** Theoretical and mathematical physicist, former CTO at Microsoft
- **Tim O'Reilly:** Publisher, open source advocate; believer that great technology needs great books
- **Jean Paoli:** One of the cocreators of the XML 1.0 standard with the W3C; now with Microsoft
- **John Patrick:** Former VP of Internet technology at IBM, now “e-tired”
- **Rob Pike:** An early developer of Unix and windowing system (GUI) technology
- **Dennis Ritchie:** Coinventor of Unix
- **Richard Stallman:** Free software movement’s leading figure; founder of the GNU Project
- **Bjarne Stroustrup:** The designer and original implementor of C++
- **Andy Tanenbaum:** Professor of computer science, author of Minix
- **Ken Thompson:** Coinventor of Unix
- **Linus Torvalds:** “Benevolent dictator” of the Linux kernel
- **Alan Turing:** Mathematician; author of the 1950 paper “Computing Machinery and Intelligence”
- **Guido van Rossum:** Author of the Python programming language
- **Ann Winblad:** Former programmer, cofounder of Hummer Winblad Venture Partners

Do come and vote online, and we’ll bring you the full results – and additional comments on the nominations – in our January 2005 issue. ☺

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, XPath 2.0, and XQuery support (including the first XSLT 2.0 and XQuery debuggers)
- SchemaAgent server
- Eclipse integration

See for yourself why XMLSpy 2005 is the standard for modeling, editing, debugging and transforming all XML technologies. Illuminate your strategy with advanced standards compliance, paradigm-shifting management 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

Also available in
the Altova XML Suite.

**Buy the Enterprise
XML Suite with
SMP from Altova NOW;
Get an Apple iPod for FREE!***

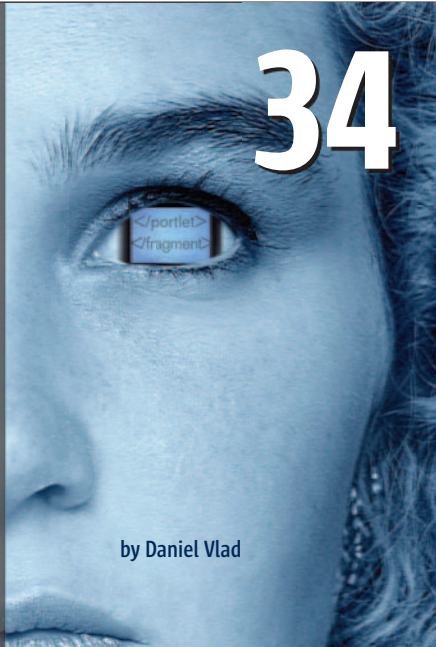
*While supplies last. Restrictions apply.
See www.altova.com for details.

JDJ contents

JDJ Cover Story

Personalize Your Web Applications

Reusable components can eliminate personalization code from your applications



34

by Daniel Vlad

Features



24

Moving to a Cluster...

by David Purcell

FROM THE GROUP PUBLISHER

The i-Technology Right Stuff

by Jeremy Geelan 3

VIEWPOINT

J2EE: A Standard in Jeopardy?

by Keith Donald 6

SPECIFICATION
EJB 3.0 Preview

Part 2: The advanced features
by Bill Burke 8

PORTLET SPECIFICATION

Understanding Portals and Portlets

Part 2: A real-world implementation
by Ken Ramirez 12

JAVA & XML

Using JAXB J2EE-Based Enterprise Applications

Part 2: Narrowing the bridge between XML and Java
by Tilak Mitra 20

SYSTEM DESIGN

Using JNDI...

...to build flexible, technology-independent enterprise systems
by Rost Vashevnik 28

CORE AND INTERNALS VIEWPOINT

Under the Hood of a J2EE Application Server

by Calvin Austin 42

Q&A

Developer Testing Is 'In'

An interview with Alberto Savoia and Kent Beck
Interview by Bill Dudney 44

DESKTOP JAVA VIEWPOINT

Sticks and Stones

by Joe Winchester 48

VIRTUAL MACHINES

Unbreakable Java

A Java server that never goes down
by Thomas Smits 54

JSR WATCH

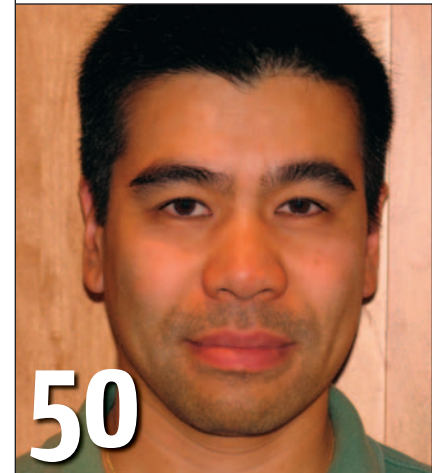
From Within the Java Community Process Program

The votes are in
by Onno Kluyt 60

@ THE BACKPAGE

Why Web Applications Can be Problematic and Unreliable

by Coach K. Wei 62



50

An Interview with Dennis Leung
VP of Oracle App Server, TopLink Development

Interview by Jeremy Geelan

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



Keith Donald

J2EE: A Standard in Jeopardy?

By now you've probably either heard about or read the analyst report from the Burton Group entitled "J2EE in Jeopardy."

In summary, the claim is J2EE as a standard is in danger due to several market forces:

1. **Market commoditization:** Open source players like Apache, JBoss, and ObjectWeb are commoditizing the platform, making it harder for vendors to profit from J2EE server licenses. If vendors can't make money on J2EE, they won't want to continue to invest in the specification.

2. **"Disruptive" technologies:**

In the last year the complexity of J2EE's programming model – EJB – has been challenged. Simpler, more productive models have emerged from within the open source community and have garnered widespread acceptance in a short time. Credited innovators here include the Spring Framework and Hibernate ORM.

The report goes on to make recommendations to mitigate risks for end users and vendors alike. In summary:

1. End users should turn over responsibility for generic J2EE infrastructure to proven providers in the open source world. You shouldn't abandon J2EE, but should consider some of the "alternative" frameworks in the open source community.
2. Vendors should focus on building a "J2EE super platform." Basically, innovate in areas where open source hasn't already preempted competition through commoditization.

As a Spring Framework user/developer and a strong believer in our value proposition and development philosophy, I

came away from all of this with mixed feelings.

On the one hand, it's great to see the work of the Spring, Hibernate, Apache, and other quality open source teams getting endorsement and credibility. The cat is out of the proverbial bag: open source is a force of innovation to be respected and certainly not underestimated. Outsourcing infrastructure to proven open source providers is a very effective strategy for companies looking to deliver working software better/faster/cheaper. Complimenting quality open source offerings with strategic commercial products and services is an effective model for infrastructure providers looking to further penetrate the market.

On the other hand, I feel the casual reader might come away with a bad spin on what's happening here. I can see it now – a corporate manager faced with a major technology investment decision happens across this article (or others like it) and concludes J2EE is in chaos. Or worse, concludes the open source community is out building flavor-of-the-month "alternative frameworks" that "reinvent the wheel" because the "standard" platform doesn't cut it. Not exactly the impression we want to make to grow the enterprise Java market.

This is where our community must step in and set that manager straight. J2EE is not in a state of chaos. There are simply more good choices for J2EE infrastructure than ever before. And from what I've experienced, there are many more J2EE success stories. Second, these "alternative" frameworks absolutely do not "reinvent the wheel in open source." They all build on standard J2EE services to improve developer productivity; they are not replacements for the platform.

from what I've experienced, there are many more J2EE success stories. Second, these "alternative" frameworks absolutely do not "reinvent the wheel in open source." They all build on standard J2EE services to improve developer productivity; they are not replacements for the platform.

– continued on page 57



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 Leevey dennis@sys-con.com
Advertising Sales Manager:
Megan Mussa megan@sys-con.com
Associate Sales Managers:
Kristin Kuhnle kristin@sys-con.com
Dorothy Gil dorothy@sys-con.com
Kim Hughes kim@sys-con.com

Editorial

Executive Editor:
Nancy Valentine nancy@sys-con.com
Associate Editors:
Jamie Matusow jamie@sys-con.com
Gail Schultz gail@sys-con.com
Assistant Editor:
Natalie Charters natalie@sys-con.com
Online Editor:
Martin Wezdecki martin@sys-con.com

Production

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

Web Services

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

Accounting

Financial Analyst:
Joan LaRose joan@sys-con.com
Accounts Payable:
Betty White betty@sys-con.com

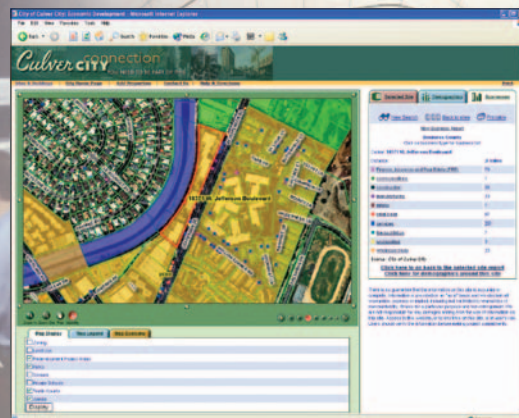
SYS-CON Events

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

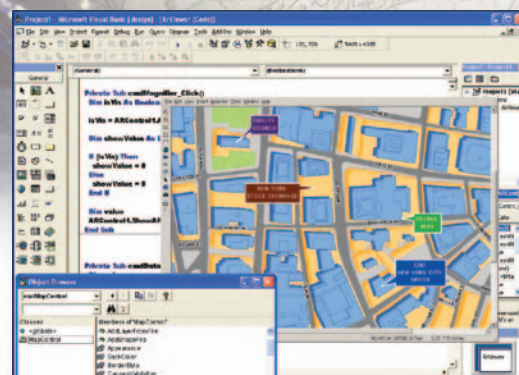
Customer Relations

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

Build Geography Into Your Applications



Web-based property management system



Using GIS components within a commercial IDE

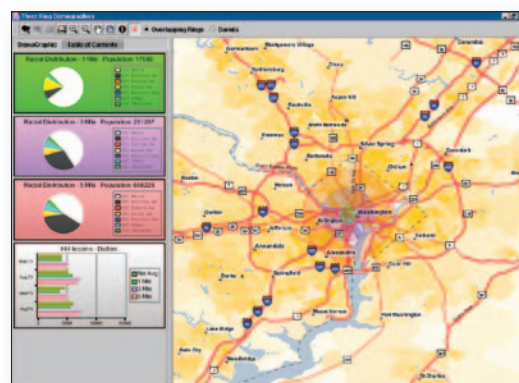
Give Your Users the Complete Picture to Help Them Make Better, Faster Decisions.

Applications that incorporate geographic information system (GIS) technology give users a visual way to analyze their data and make more informed decisions. With ESRI® developer solutions, you can quickly and cost-effectively bring geography and mapping capabilities into your applications, regardless of whether you are building desktop, client/server, mobile, or Web applications.

ESRI developer solutions enable you to

- ▶ Quickly and cost-effectively integrate GIS capabilities into your new and existing applications.
- ▶ Select the developer tools that fit best with your architecture (ESRI's developer products encompass GIS components, servers, and Web services).
- ▶ Use the development environment of your choice, including Java™, .NET, COM, and C++, and deploy applications on a variety of platforms.
- ▶ Access and manipulate data in multiple formats.

To learn more about the ESRI developer solutions that are right for you, visit www.esri.com/develop.



Population demographics analysis application



1-888-288-1277

www.esri.com/develop

info@esri.com

EJB 3.0 Preview

by Bill Burke

The advanced features Part 2

Last month's article on EJB 3.0 (Vol. 9, issue 11) focused primarily on the basic features of the specification. Part 2 dives much deeper into the specification to talk about more advanced features like dependency injection, dependent objects, secondary tables, and inheritance.

Dependency Injection

Dependency injection is the opposite of `JndiContext.lookup()`. The idea of dependency injection is that objects and services specify which resources and configuration values they require, and their container automatically injects these values. Dependency injection is supported for any session bean type and will also be supported in the greater J2EE specification in items such as servlets. This approach requires no JNDI lookup at all and can greatly simplify code. Let's look at an example.

```
@Stateful
public class ShoppingCartBean implements
ShoppingCart
{
    @Inject private UserTransaction userTx;
    @Inject private EntityManager entity-
Manager;

    private Petstore store;

    @EJB(name="petstore") public void
```

```
setPetstore(Petstore store) {
    this.store = store;
}
```

When an instance of `ShoppingCartBean` is allocated, the EJB container will look up the `UserTransaction` service and set the `userTx` variable. It will also get a reference to the EJB whose `ejbName` is "petstore" and call the `setPetstore()` method. Values can be injected either with an explicit field set or by calling a setter method. A great side effect of injection is that it becomes possible to test beans outside the context of a container.



`@Resource` is another annotation for injecting things like `DataSources` and `JMS` connections.

Dependent Objects

The EJB 3.0 specification defines a full object/relational mapping for dependent value classes. You can map the properties of an aggregated value object in your entity to specific

columns of the entity's table.

```
@DependentObject(access=AccessType.PROPERTY)
public class Address implements java.
io.Serializable {

    private String street;
    private String state;
    private String city;

    public String getState() { return state;
}
...
}
```

Your dependent value class can either have its properties defined as `get/set` methods or directly as fields. Next, define the mapping of a `@DependentObject` within your entity bean (see Listing 1).

Multi-Table Mappings

Many application developers find it necessary to map an entity bean to multiple tables within a database, especially when they have to map objects to a legacy data schema. EJB 3.0 provides mappings for this using the `@SecondaryTable` annotation.

```
@Entity
@SecondaryTable(name="ADDRESS", join={@Join
Column(name="address_id")})
public class Customer {
...
    @Column(name = "street", secondaryTable
= "ADDRESS")
```



Bill Burke is chief architect of JBoss Inc., member of the EJB3 expert group, and co-author of the *JBoss 4.0 Workbook* in O'Reilly's Enterprise JavaBeans, 4th Edition.

bill@jboss.org

“The EJB 3.0 specification defines a full object/relational mapping for dependent value classes”



Java that's ready for anything.

Borland® JBuilder® 2005, the newest edition of the #1 Java™ IDE in the world. With the most advanced feature set ever, ideal for enterprise-class projects. Complete command and control for JavaServer™ Faces (JSF™), JavaServer™ Pages (JSP™), servlets, Enterprise JavaBeans™ (EJB™), Web Services, Struts, XML, Swing, database applications, mobile applications, and more. Take your next project up to the next level. And enjoy peak performance.

- Import project source from any IDE or editor
- JSF-enables existing Web applications
- New distributed refactoring for better teamwork
- Customizable code editor with CodeInsight™ and ErrorInsight™
- Two-way visual Struts designer
- Supports Apache™ Axis 1.2, SOAP, WSDL, UDDI, and WSIL
- JSP™ tag library/framework support
- Tight integration with Borland® Enterprise Server, BEA WebLogic™ Server, IBM® WebSphere®, and more

go.borland.com/j6

Made in Borland® Copyright © 2004 Borland Software Corporation. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. • 22953.3

Borland®

```

public String getStreet()
{
    return street;
}
...

```

The `@SecondaryTable` is defined as a class annotation and specifies the table's name as well as the columns to use to join the main and subtable together. The `@JoinColumns` of the secondary table must map directly to the primary key of the entity.

To map a specific property to the

SINGLE_TABLE Strategy

The `SINGLE_TABLE` specifies that there should be one and only one table per class hierarchy. This table should have a column for each unique field for every class in the hierarchy. The table must have an additional column that identifies the object's type. By default, its type is a string with the default value being the fully qualified class name of the object stored. The `@DiscriminatorColumn` maps the object identity to a specific database column. The

All entities that subclass from `Animal` can be queried polymorphically:

```

Query query = entityManager.
createQuery("from Animal a where
a.averageWeight > 10");

```

This query could return an instance of a `Dog`, `Cat`, `Elephant`, whatever entities that are currently defined in the `Animal` hierarchy.

Finally Usable

EJB 3.0 finally makes EJB persistence a reality. With EJB 2.1 entities you continually had to escape to direct JDBC, rely on vendor proprietary extensions, or move to an entirely different object/relational mapping strategy altogether. Features such as a full object relation mapping including a fully featured query language, inheritance, secondary tables, and dependent objects finally make the EJB specification.

With the use of annotations, XML deployment descriptors that have long been the bane of EJB developers can be completely removed if so desired by the developer.

All in all, EJB has come a long way since the 1.0 days and is morphing itself to the specifications of the community for which it was written. ☺

References

- JSR 220: Enterprise JavaBeans 3.0: www.jcp.org/en/jsr/detail?id=220
- JBoss, EJB 3.0: www.jboss.org/ejb3

“EJB has come a long way since the 1.0 days”

secondary table, specify the `secondaryTable` annotation member value from the `@Column` annotation.

Entity Inheritance

Another missing feature in EJB 2.1 is the ability to support inheritance and map a complex class hierarchy to a relational database. EJB 3.0 supports inheritance and polymorphic queries. Three types of inheritance mapping strategies are supported: one table per class hierarchy (`SINGLE_TABLE`), a join table per subclass (`JOINED`), and a distinct table per class (`TABLE_PER_CLASS`). Only `SINGLE_TABLE` is required by the EJB 3.0 specification, so we'll focus on an example covering that.

`SINGLE_TABLE` strategy is the optimal strategy for performance as the persistence engine doesn't have to do any complex joins when loading such an object. For example, say we have an `Animal` superclass entity bean, and a `Dog` subclass. The Java code would look like Listing 2.

The table mapping would be one gigantic table:

```

create table Animal (
    ID Number,
    TYPE varchar(255),
    AVG_WEIGHT Number,
    BREED varchar(255)
);

```

Listing 1

```

@Entity
public class Customer {
    private Address address;
    ...
    @Dependent({
        @DependentAttribute(name="street", column={@Column("STREET")}),
        @DependentAttribute(name="city", column={@Column("CITY")}),
        @DependentAttribute(name="state", column={@Column("STATE")})
    })
    public Address getAddress() { return address; }
    ...
}

```

Listing 2

```

@Entity
@Table(name="Animal")

```

```

@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="TYPE")
public class Animal
{
    @Id private int id;

    @Column(name="AVG_WEIGHT")
    private int averageWeight;

    ...
}

@Entity
public class Dog extends Animal
{
    @Column(name="BREED")
    private String breed;

    ...
}

```

**IMPORT PEERS.*;
IMPORT EXPERTS.*;
IMPORT YOU.*;**

**// EXPORT DEADLOCK
// GO TO SDN.SAP.COM**



You're stuck. You need answers. Maybe you have a solution to share with other SAP developers or a question for an SAP insider. Get the experts, partners and your colleagues to weigh in. Now there's a single collaborative destination where you can all converge: SAP® Developer Network. Nowhere else can you join a spirited discussion forum, download a trial of the latest SAP Web Application Server, take an e-learning course, and, in general, keep us on our toes.

// JOIN IN AT SDN.SAP.COM

Understanding Portals and Portlets

A real-world implementation Part 2

by Ken Ramirez

In the November issue of *JDJ* (Vol. 9, issue 11) I explained the theory behind the JSR 168 (Portlet Specification) from an academic perspective. The specification provides the infrastructure, classes, interfaces, and JSP tags for building applications that can be pieced together from a handful of off-the-shelf or custom portlets. This time around, I provide you with a real-world implementation that utilizes the knowledge you picked up from Part 1 of this series.

Along the way, you'll learn how to properly install Pluto (the Portlet Reference Implementation you can use to test your portlets) and learn about some pretty cool tools as well.

A Quick Refresher

Last month, we learned that portlets are similar to servlets, except they aren't allowed to use several HTML control tags such as the `<body>` tag. This is because the portal page provides these controlling tags, whereas the portlets provide the tags necessary to complete their own work.

Portlets also share the application context with servlets and JSPs and can even include the output of another servlet or JSP as part of their content. To offer the user the ability to customize portlets, there are special Window States (such as normal, minimized, and maximized) and Portlet Modes (such as Edit, View, Help) that can be controlled by the end user. Furthermore, the portlet can use these states and modes to determine what content it needs to show the user at any given point. User's actions are received within a portlet in the form of both action and render methods. Action methods respond to the user's interaction with the portlet, and render methods are called to paint the output of the portlet.

Later in this article, we'll see how we can put all of these features and settings to use in our example portlet. However,

before we can dive into the code, we need to ensure that our environment is ready to test a portal page and its contained portlets.

Required Downloads

Given the fact that Pluto is very Mavenized these days, you'll need to download Maven (a tool that allows you to build and deploy your project based on a Project file). In addition, you'll need a servlet engine to run Pluto (since Pluto runs as a servlet). I suggest Tomcat (preferably version 5) as the servlet engine.

If you're developing on a Windows machine, the Tomcat setup is pretty easy. I downloaded Tomcat from the Apache Web site using <http://jakarta.apache.org/tomcat/>.

Make sure you extract the Tomcat binaries into its own directory and be sure you have the `JAVA_HOME` environment variable pointing to your Java installation directory path. Once Tomcat is installed, try your hand at Maven. Download and install Maven from the Apache Web site as well.

Define an environment variable named `MAVEN_HOME` and point it to the Maven root directory. You may also want to add Maven's bin directory to

your path, so you can easily execute Maven from any command prompt window in any directory. If all goes well, you should be able to run Maven with the `-v` option to view its version information.

Finally, it's time to locate and download Pluto. At the time of this writing, the latest version of Pluto is 1.0.1, and is in a Subversion Source Code Management (SCM) system. If you visit <http://cvs.apache.org/viewcvs.cgi/portals/pluto/trunk/?root=Apache-SVN>, you'll find the trunk.

You can either use the Web interface to surf each package and download them directly (which is very cumbersome), or you can download and use Tortoise for Subversion, which adds context options to Windows Explorer in order to assist you in downloading content from various Subversion repositories across the Internet. The Tortoise installation and documentation is located at <http://tortoisesvn.tigris.org>.

Tortoise provided Windows Explorer with new context menu items as shown in Figure 1.

Next, you'll need to create a new directory and download Pluto. Right-click in the newly created directory while in Windows Explorer to bring up the context menu and select the option marked "Checkout...". You'll need to specify the complete path to the Subversion repository, which is mentioned on the Apache.org Web site as follows <http://svn.apache.org/repos/asf/>.

A Subversion repository is similar to CVS, but provides added features and is becoming extremely popular. Many sites are changing their repositories from CVS to Subversion (and the Apache group is no exception).

I simply added the remainder of the URL to locate the Pluto project as shown in Figure 2.



Ken Ramirez has 17 years of experience providing development services, consulting, and training to companies (both large and small) throughout the United States. He consults in various market industries including finance, insurance, computer-aided design, community portals, and automobile. Ken's Java expertise includes J2EE, XML, portals, UML, and many open source technologies. His latest venture is the www.TheJavaThinkTank.org community portal site.

kramirez@TheJavaThinkTank.com

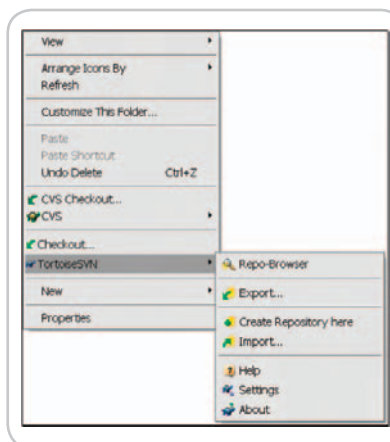


Figure 1 New options added by Tortoise

Once you download Pluto's source files you'll have a complete directory tree with the necessary files needed to build Pluto. You can always update Pluto with newer changes by opening the context menu offered by Tortoise. You'll notice that the menu will change to reflect new choices available now that you have downloaded Pluto.

To run Maven against the Pluto file, you'll need to perform the following tasks:

1. Copy the build.properties.sample file to build.properties.
2. Modify the build.properties and specify the location and version of Tomcat:

```
maven.tomcat.home=/tomcat-5.0.27
maven.tomcat.version.major=5
```

3. Run the maven fullDeployment command.

The fullDeployment operand tells Maven to run this task within the project file. The task has been provided with everything needed to download any necessary or missing projects, compile Pluto, and deploy Pluto.

Now, you're ready to access Pluto for the first time (it comes with a test portal). First, you'll need to run Tomcat. Once Tomcat has finished loading, you can access the portal by surfing over to <http://localhost:8080/pluto/portal>. You should be presented with the page shown in Figure 3.

If you click on the Test portal, you'll be able to test out much of what a JSR 168 portlet has to offer. You'll see two instances of the same portlet host on a portal page with two columns (one for each portlet). The source code can be located at [pluto-installation-dir]/test-suite.

The ImageViewer Portlet

When I thought about writing the example for this article, I wanted to come up with an idea that was simple enough to understand, yet realistic enough to give you a starting point for building your own real-world portlets. (The source code for this article can be downloaded from www.sys-con.com/java/sourcecec.cfm.) The result is a portlet that utilizes many JSR 168 features to provide an end user with a portlet that has two portlet modes: view and edit.

The view mode allows the end user to specify an image stored in a subdirectory named images on the server side (relative to the portlet's context root), and displays the image within the portlet on the way back to the browser. After entering a name and clicking the "Show Image" button, the user is presented with the result as illustrated in Figure 4.

You'll notice that there is some text in the portlet that specifies, "color = white". White is the default color chosen for the portlet's background. However, this can be changed by entering the Edit mode of the portlet, which is achieved by clicking on the "edit" link on the portlet's title bar. When you do so, you're presented with the output shown in Figure 5.

If you change the selection to "Blue" and press the "Change Color" button, then go back to View mode and select an image, you'll notice that the background now reflects the new background color.

The ImageViewer Source Code Explained

Now that you've seen the ImageViewer portlet from the end-user's perspective, let's take an in-depth look at the source code.

The first thing I did was to set up my directory structure, which resembles the image shown in Figure 6.

I knew that I would be building the files listed in Table 1.

The Portlet.XML Deployment Descriptor File

Let's look at the portlet.xml file, which contains the various descriptors used to deploy the portlet properly (see Listing 1).

The first part of the file includes the standard customary XML tag used to identify the file and the schema used to validate the file's content. Next, we see the root portlet tag. Within this tag are several other tags including description, portlet-name, display-name, portlet-class, init-param (of which there are three), supports (for mime types), locale, and portlet-info. The most important of these tags are the portlet-name and the portlet-class tags.

The portlet-name provides a name that can be used internally or programmatically to reference the portlet, while the portlet class provides the classpath

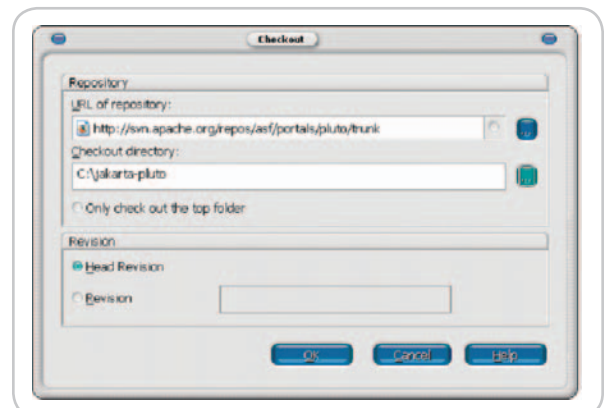


Figure 2 The completed Checkout dialog box in order to retrieve Pluto

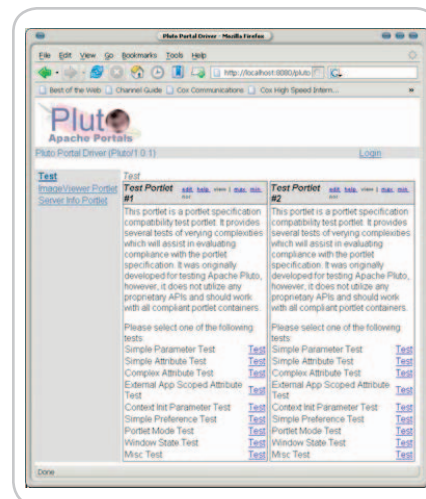


Figure 3 Pluto running in Tomcat

File	Location	Description
portlet.xml	ImageViewer/war/WEB-INF	Contains the portlet's deployment descriptors
edit.jsp	ImageViewer/war/jsp	Generates the HTML needed for the Edit mode of the portlet
view.jsp	ImageViewer/war/jsp	Generates the HTML needed for the View mode of the portlet
various image files	ImageViewer/war/images	Contains any number of images you wish to view through the portlet
ImageViewerPortlet.java	ImageViewer/src/com/thejavathinktank/portlets	Contains the source code for the portlet
build.properties	ImageViewer root directory	Contains any properties used in the build process
build.xml	ImageViewer/build	Contains the Ant build script

Table 1 The list of files built to support the ImageViewerPortlet



used by the portlet container to load the appropriate class and create instances from it.

The description tag can be used by development tools to provide a description of what the portlet does, whereas the display name can be used to provide the developer with a unique name that (hopefully) differs from other portlets in the portlet catalog. As portals become more popular, you'll start to see drag-and-drop support within the various development environments, allowing developers to drag portlets onto a portal page and drop them into the page. The display name will become very important, showing the developer which space is occupied by which portlet.

I included three different init-param values that provide the portlet with the names of the JSPs used for its View and Edit modes and the default background color of white.

The supports tag provides the mime-type tag, which tells the portlet container that the portlet will provide HTML for the various views and that two different modes are supported: View and Edit. The supported locale tag specifies that our chosen language is English.

Finally, the deployment descriptors provide the title tag, used by default for providing a title for the portlet in the portal page; the short title, which can be

used to provide a shorter version of the title (for use in WAPs or other devices that have a smaller screen display area); and keywords, which are used to search for portlets within development tools. The search can, for example, narrow down the catalog of portlets to a few portlets that support stock screening or image manipulation, or whatever other functionality you're interested in providing your end users with from your portal pages.

The ImageViewerPortlet Java Source File

The source code for the ImageViewerPortlet.java file starts off by defining the package name and importing the necessary packages and/or classes. Next, it names the class and provides the methods. There are no data members in this class (see Listing 2).

The first method we'll look at is `init()`. This method doesn't have much to do in this case, but I wanted to assure you that it is in fact called. Therefore, I simply added an output message to the console.

Now it's time to meet the `processAction()` method. Based on the retrieved portlet mode, the portlet determines whether it's in View mode or Edit mode. If it is in View mode, this method simply has to copy the present parameters (passed from the user's browser, which includes the image name the user wishes to see) into the response's render parameters so they can be picked up by the `doView()` method. If the portlet is in Edit mode, it retrieves the current background color from the request (this is always passed in the JSP as you'll soon see), and places it in the `PortletSession` (where it can be picked up by the JSP page). I could have used portlet preferences for the background color (along with a validator class), but I chose to keep the example simple.

Next, we come to the `doView()` method. This method retrieves the `PortletSession` object, which can be used to then retrieve the `bgColor`. This is just one of many ways to pass data around. I simply wanted to use as many techniques as possible without overwhelming you.

If the background color has not been set into the session, it's set at this point by retrieving the default color from the portlet's configuration object using the

`getInitParameter()` method.

Next, I set the mime type for the view mode and dispatch to the JSP page (including its output in the final HTML sent to the browser).

Finally, the `doEdit()` method appears in the listing. This one is pretty simple and resembles the bottom half of the `doView()` method. The only difference is that it dispatches to the Edit JSP instead of the View JSP.

The View JSP

The `view.jsp` file provides the source code in Listing 3. The top portion of this file includes the portlet tag library. One of the most important steps you must take (before you can use any of the portlet internal objects) is to call `portlet:defineObjects` tag, which is provided by the tag library. Next, it creates a few variables to hold the action URL, the image filename the user specified, and the background color (retrieved from the session object).

After creating the variables, the HTML provides a string containing the current color, a form to include the controls used by the user to interact with the portlet, and the submit button.

If a filename is provided, the name is used with some special methods to conjure up a complete path to the actual image file. You *must* use the `renderRequest.getContextPath()` method to create a legitimate path. Don't assume that you could simply use the relative path and it will work.

Finally, the newly created path is used in the image tag to display the image within the portlet.

The Edit JSP

The code for the Edit JSP isn't much different from that of the View JSP. It only differs in its content, which should be very self-evident (if you know HTML at least at a basic level). See Listing 4 to view the source code for the `edit.jsp` file.

Deploying to Pluto

To build the source code, I created an Ant file. Once you have compiled the code, you can deploy it to the Pluto portal using the following commands:

```
cd\jakarta-pluto
maven deploy -Ddeploy=/ImageViewer/target/
ImageViewer.war
```

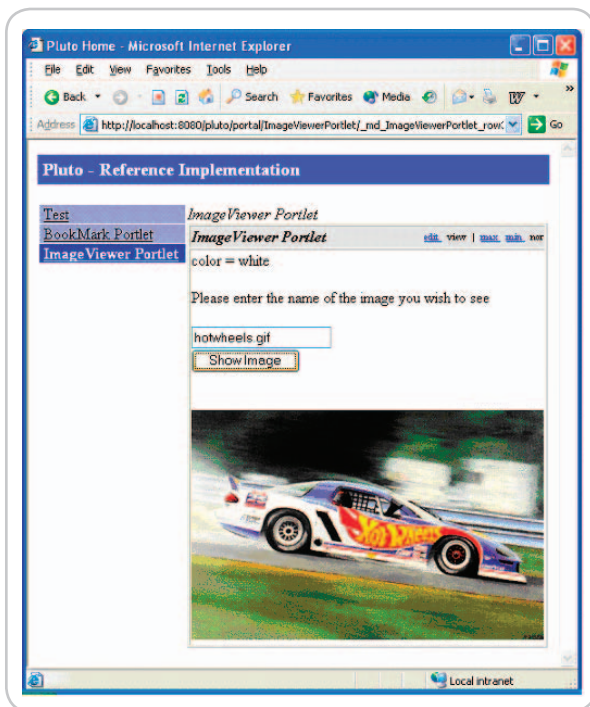
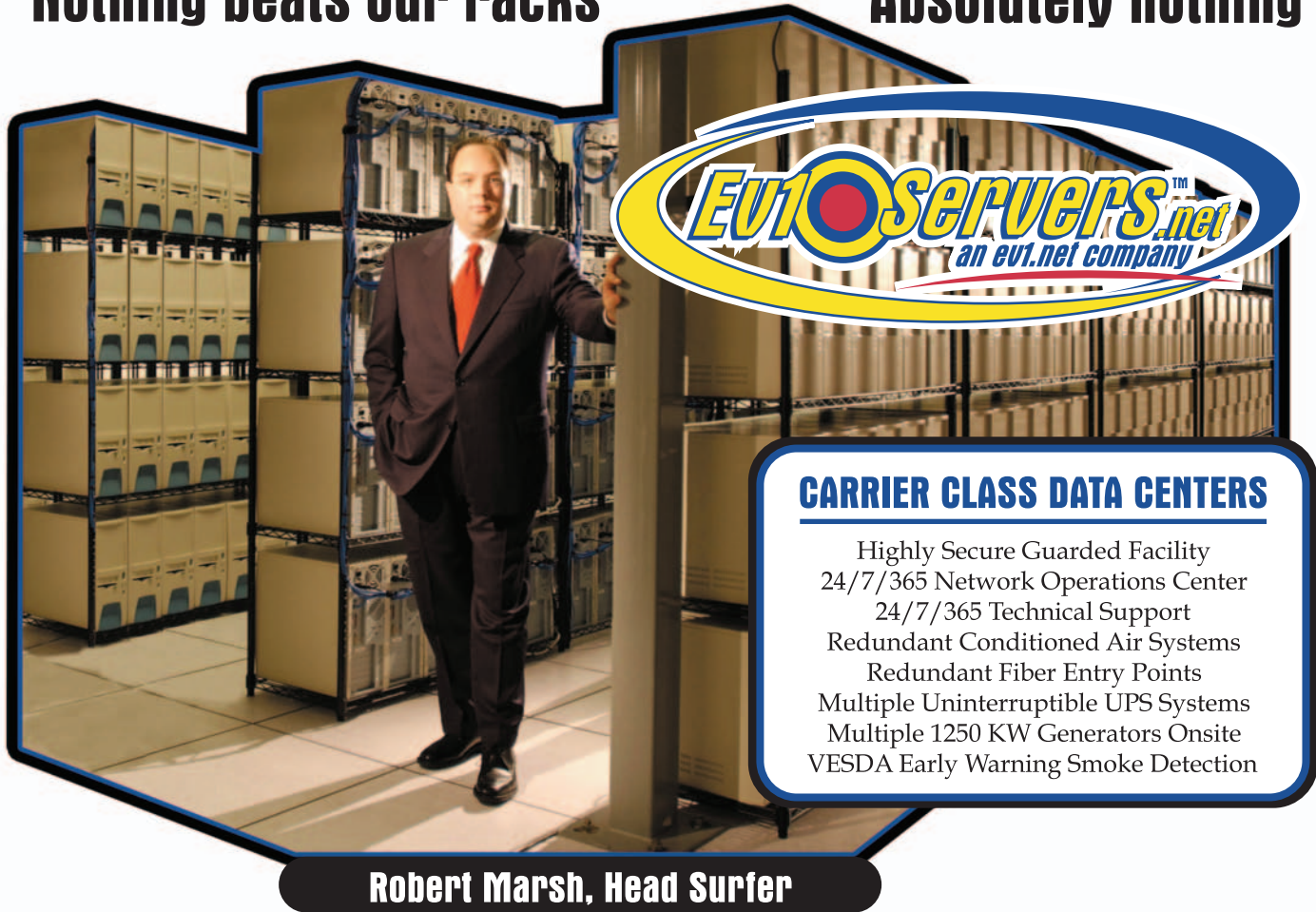


Figure 4 ImageViewer portlet after specifying an image name

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.

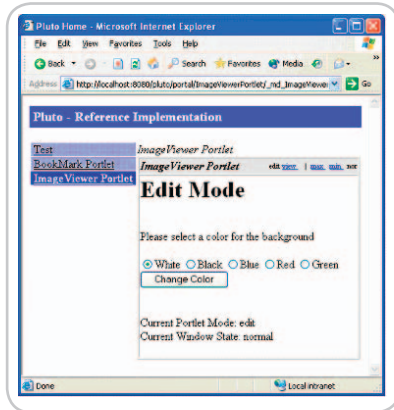


Figure 5 Edit mode of ImageViewer portlet

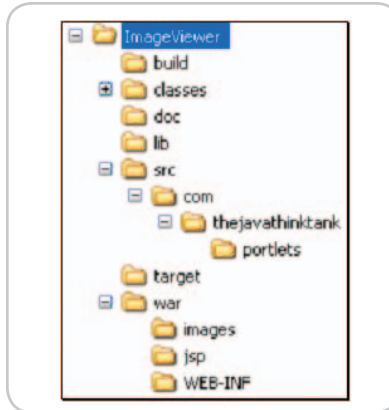


Figure 6 Project directory structure

There are still a few more steps you have to follow in order to view the portlet within Pluto. You need to modify the Portlet Entity Registry and the Page Registry files. These can be located at:

```
[tomcat-home]/webapps/pluto/data/portletentityregistry.xml
[tomcat-home]/webapps/pluto/data/pageregistry.xml
```

The portlet entity registry file requires that you specify an application and a portlet ID for your new portlet. The application ID must be unique. The portlet entity registry file also needs to know the name of the portlet so that it can go out and find it in the webapps path. Furthermore, this information is used to map the portlet name to the appropriate

classpath for loading the class. The following are my additions to the portlet entity registry file:

```
<application id="6">
  <definition-id>ImageViewer</definition-id>
  <portlet id="1">
    <definition-id>ImageViewer.
    ImageViewerPortlet</definition-id>
  </portlet>
</application>
```

The page registry provides Pluto with the layout information for your portlet. The names used in the fragments must be unique as in my example:

```
<fragment name="ImageViewerPortlet"
type="page">
```

```
<navigation>
<title>ImageViewer Portlet</title>
<description>...</description>
</navigation>

<fragment name="row3" type="row">
  <fragment name="col3" type="column">
    <fragment name="p4" type="portlet">
      <property name="portlet" value="6.1"/>
    </fragment>
  </fragment>
</fragment>
```

In the above example, I simply told Pluto that I will need one row by one column. Within that row/column, I want it to display the portlet identified by "6.1". The number "6" is the application ID and "1" is the portlet ID.

After you edit and save the two XML files, restart Tomcat and you should now see the Portal page link named "Image Viewer".

Summary

Although I've tried to provide you with as much information and features as I could, the only true way to learn how to build the JSR 168 portlets is to jump right in and do it. You should be able to use this code as the foundation for other portlets. Remember, if you do decide to start creating portlets, try to make them self-contained units of work. That way, they can be shared across projects, teams, and possibly even companies. ☺

Listing 1: portlet.xml file

```
<portlet>
<description>Image Viewer Portlet</description>

<portlet-name>ImageViewerPortlet</portlet-name>

<display-name>Image Viewer Portlet</display-name>

<portlet-class>com.thejavathinkank.portlets.ImageViewerPortlet</
portlet-class>

<init-param>
  <name>jspView</name>
  <value>/jsp/view.jsp</value>
</init-param>
<init-param>
  <name>jspEdit</name>
  <value>/jsp/edit.jsp</value>
</init-param>
<init-param>
  <name>bgColor</name>
  <value>white</value>
</init-param>
```

```
<supports>
  <mime-type>text/html</mime-type>
  <portlet-mode>VIEW</portlet-mode>
  <portlet-mode>EDIT</portlet-mode>
</supports>
<supported-locale>en</supported-locale>

<portlet-info>
  <title>ImageViewer Portlet</title>
  <short-title>ImageViewer</short-title>
  <keywords>Images, Viewer</keywords>
</portlet-info>
</portlet>
</portlet-app>
```

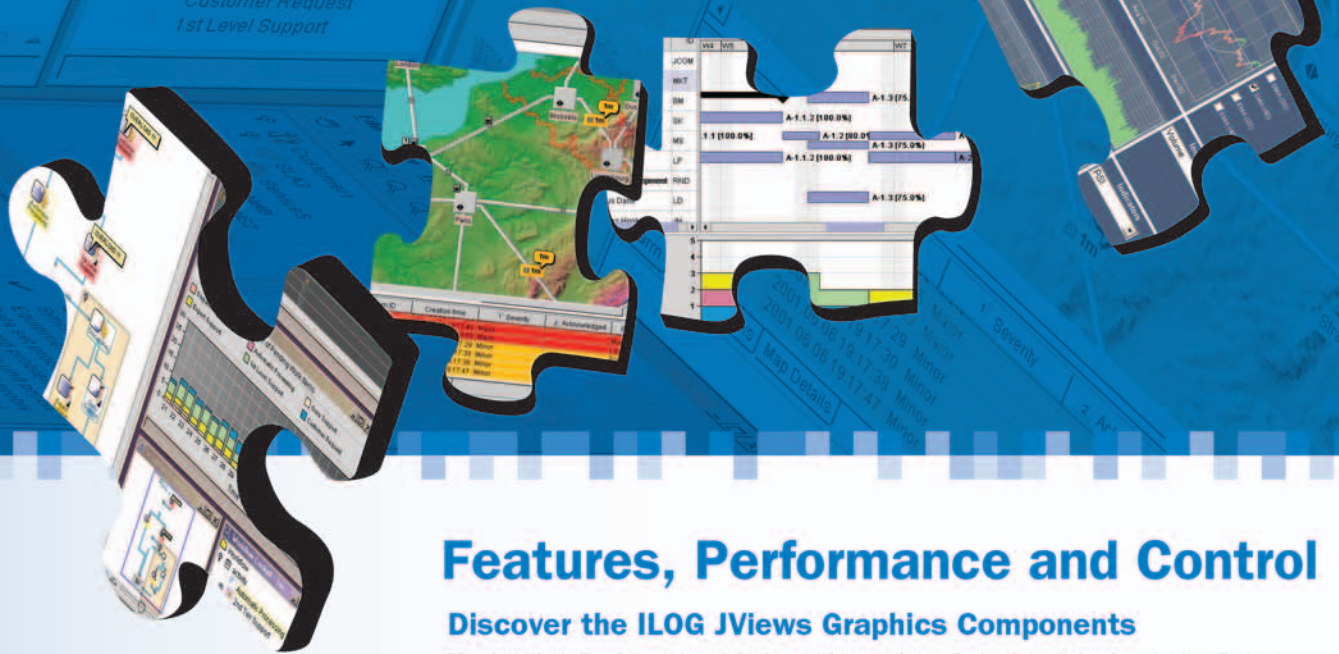
Listing 2: The ImageViewerPortlet Java class

```
package com.thejavathinkank.portlets.imageviewerportlet;

import javax.portlet.*;
import java.io.IOException;

public class ImageViewerPortlet extends GenericPortlet
{
  public void init() throws PortletException
  {
```


Get the complete picture



Features, Performance and Control

Discover the ILOG JViews Graphics Components

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

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

Quickly and easily build:

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

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



Changing the rules of business™



DESKTOP



CORE



ENTERPRISE



HOME

```

        System.out.println("Entered ImageViewerPortlet.init() meth-
od");
    }

    public void processAction(ActionRequest request, ActionResponse
response)
        throws PortletException, java.io.IOException
    {
        PortletMode pm = request.getPortletMode();

        System.out.println("PortletMode = " + pm);

        if (pm.equals(PortletMode.VIEW))
        {
            System.out.print("Request Image file is: ");
            System.out.println(request.getParameter("filename"));
            response.setRenderParameters(request.getParameter-
Map());
        } else if (pm.equals(PortletMode.EDIT))
        {
            String bgColor = request.getParameter("bgColor");
            PortletSession ps = request.getPortletSession();
            ps.setAttribute("bgColor", bgColor);
            System.out.println("New color = " + bgColor);
        }
    }

    public void doView(RenderRequest request, RenderResponse
response)
        throws PortletException, IOException
    {
        // Assign bg color if it doesn't exist
        PortletSession ps = request.getPortletSession();
        String bgColor = (String) ps.getAttribute("bgColor");
        if (bgColor == null)
        {
            bgColor = getPortletConfig().getInitParameter("bgColo
r");
            System.out.println("Setting default bg-color to " +
bgColor);
            ps.setAttribute("bgColor", bgColor);
        }

        response.setContentType("text/html");

        String jspName = getPortletConfig().getInitParameter("jspV
iew");

        PortletRequestDispatcher rd = getPortletContext().getRe-
questDispatcher(
            jspName);

        rd.include(request, response);
    }

    public void doEdit(RenderRequest request, RenderResponse
response)
        throws PortletException, IOException
    {
        response.setContentType("text/html");

        String jspName = getPortletConfig().getInitParameter("jspE
dit");

        PortletRequestDispatcher rd = getPortletContext().getRe-
questDispatcher(
            jspName);
    
```

```

        rd.include(request, response);
    }
}

```

Listing 3: The view.jsp file

```

<%@ page session="false" %>
<%@ page import="javax.portlet.*"%>
<%@ page import="java.util.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix='portlet'%>
<portlet:defineObjects/>

<%
    PortletURL url = renderResponse.createActionURL();
    String filename = (String)renderRequest.getParameter("filename");
    PortletSession ps = renderRequest.getPortletSession();
    String bgColor = (String)ps.getAttribute("bgColor");
    %>
color = <%=bgColor%>
<form method="post" action="<%=url.toString()%>" style="background-
color: <%=bgColor%>">
<p>Please enter the name of the image you wish to see</p>
<input type="text" id="filename" name="filename"/>
<br />
<button type="submit">Show Image</button>
</form>
<br />
<%
    filename = renderResponse.encodeURL(renderRequest.getContext-
Path()+"/images/"+filename);
    %>


```

Listing 4: The edit.jsp file

```

<%@ page session="false" %>
<%@ page import="javax.portlet.*"%>
<%@ page import="java.util.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix='portlet'%>
<portlet:defineObjects/>

<h1>
Edit Mode
</h1>
<br>
Please select a color for the background<br>
<%
    PortletURL url = renderResponse.createActionURL();
    PortletSession ps = renderRequest.getPortletSession();
    String bgColor = (String)ps.getAttribute("bgColor");
    %>
<form method="post" action="<%=url.toString()%>">
<input name="bgColor" type="radio" value="white"
    <%=bgColor.equals("white") ? "CHECKED" : ""%>/>White
<input name="bgColor" type="radio" value="black"
    <%=bgColor.equals("black") ? "CHECKED" : ""%>/>Black
<input name="bgColor" type="radio" value="blue"
    <%=bgColor.equals("blue") ? "CHECKED" : ""%>/>Blue
<input name="bgColor" type="radio" value="red"
    <%=bgColor.equals("red") ? "CHECKED" : ""%>/>Red
<input name="bgColor" type="radio" value="green"
    <%=bgColor.equals("green") ? "CHECKED" : ""%>/>Green
<br/>
<button type="submit">Change Color</button>
</form>
<br>
Current Portlet Mode: <%=renderRequest.getPortletMode()%><br>
Current Window State: <%=renderRequest.getWindowState()%><br>
<br>

```

NEW
VERSION

InstallShield®



- ✓ *Linux*
- ✓ *Unix*
- ✓ *Mac OS X*
- ✓ *Windows*

We do it all. Introducing InstallShield 10.5, the industry's choice for the strongest, most reliable installations that can run everywhere.

Download Now!

www.installshield.com/JDJ

Special Offer!

Receive expert training with purchase.
See Web site for details.

Using JAXB in J2EE-Based Enterprise Applications

by Tilak Mitra

Narrowing the bridge between XML and Java Part 2

In Part 2 of this two-part series (Part 1 appeared in Vol. 9, issue 4) I shall try to construct an XML Schema, take you through the steps required to convert an XML document into its corresponding Java classes and interfaces, and also show how to generate an XML document (by using the generated Java classes and interfaces) from a Java object tree, in a programmatic fashion.

A Brief Recap

Part 1 introduced the fundamental concepts of Java Architecture for XML Binding (JAXB) and gave some insights into how it can be used in a typical J2EE-based enterprise application.

The power of JAXB comes from the fact that it removes the developer from the shackles of arduous XML-to-Java (and vice versa) conversion. Its feature-rich specification allows a Java developer to incorporate Java-specific programming constructs as part of the XML Schema. Among the rich set of features, the specification class allows the addition of Java packages, Javadoc comments, Collection classes, etc., to the XML Schema. Once configured, this becomes part of the Java classes and interfaces that are generated by the JAXB compiler.



Tilak Mitra is a Senior IT architect at IBM. He specializes in mid-to-large-range enterprise and application architectures based on SOA, J2EE, MQ, and other EAI technologies.

tmitra@us.ibm.com

Example Scenario

We are going to work with an example of a simple library. An oversimplified library may be modeled as a collec-

tion of books. Each book has a list of attributes (title, author, ISBN code, etc.). Books may be added or removed from a library.

Referring to Listing 1, the complex-Type Book has three attributes while the complexType Library is composed of a collection of elements of type Book. BookItem is declared as an element of type Book while CityLibrary is declared as an element of type Library. Any XML document that conforms to this XML Schema can either have a CityLibrary or a BookItem as the root element. A CityLibrary element may contain one or more BookItem elements and any BookItem element contains three attributes. Any reference to elements within the XML Schema is qualified by a namespace – domainObjects in this scenario.

Once the XML Schema is defined, the first step is to use your favorite editor to compile the schema using the JAXB compiler. This produces the Java classes and interfaces in the desired package(s). Listing 2 provides the output of running the JAXB compiler on our schema file.

The compiler can be run with the following command:

```
xjc.bat -p <package name> -d <working directory>
```

where <package name> is the package where the Java artifacts are to be generated and <working directory> is the directory in the file system where the packages are going to be generated.

(Note: For a Unix-based system, the compiler script is called xjc.sh.)

Figure 1 illustrates the list of Java artifacts that are generated after running the schema through the JAXB compiler.

Once this is created, the developer is ready to harness the real potential of JAXB. The developer can now interpret any XML file that conforms to the XML Schema (from which the above Java artifacts were generated). The process of interpretation is known as unmarshalling (the creation of a Java object tree from an XML document). You can also use the generated classes to programmatically create an XML document. This process is known as marshalling (the creation of an XML document from the Java objects).

The first step for the developer is to obtain a reference to the JAXBContext object. This object has methods to retrieve references to the Marshaller and the Unmarshaller objects.

The Marshaller object instance is used in any subsequent marshalling process while the Unmarshaller object instance is used in any subsequent unmarshalling process.

Listing 3 is a sample Java class that illustrates how the JAXB framework can be used. The method createContext() creates instances of the Marshaller and Unmarshaller classes. (Listings 3–5 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

Unmarshalling

In unmarshalling, an XML document (as in Listing 4) is accepted as

“The power of JAXB comes from the fact that it removes the developer from the shackles of arduous XML-to-Java (and vice versa) conversion”

the input. This XML document is interpreted by the program and corresponding Java classes are instantiated.

The root element of the XML document is CityLibrary, which contains two Books. Referring to Listing 5, the method unmarshallIt creates the root element CityLibraryImpl from the XML document. This generated class has methods to iterate through the list of contained BookImpl objects.

It's up to the application requirements from this point – how to use the data that is now represented as simple Java objects. There is no Java XML parsing that is required to interpret the XML document artifacts. The simple example that is shown here iterates through the list of books and prints out the title and author attributes in each iteration.

Marshalling

In marshalling, the Java objects (generated by the JAXB compiler) are used to create an XML document whose contents conform to the XML Schema (see Listing 5).

A creational class called ObjectFactory (generated by the JAXB compiler) is used to instantiate any class in the Java object tree. A careful look into the classes (in the Java object tree) illustrates that the element containment structure in the XML Schema is represented as contained object references (using the Java Collections Framework). Attributes, however, have getter and setter methods in the Java object's representation of the contained element.

Referring to Listing 3, the method marshallIt first creates an instance of the root element of a sample XML document (instance of CityLibraryImpl). It then obtains the collection inside the root element that's used to add the nested elements (BookImpl in this case). Instances of BookImpl are created using the ObjectFactory class. Attributes of instances of BookImpl are added using the setter methods of the corresponding BookImpl instances. Once an instance of BookImpl is created and its attributes set, the instance is added to the collection of the container CityLibraryImpl instance. This way multiple BookImpl instances can be added to the CityLibraryImpl instance. Once the object structure is created, obtaining the structure's correspond-

ing XML representation is a simple matter of invoking a method on the Marshaller instance, and passing the reference of the CityLibraryImpl (root element) as a parameter.

Final Thoughts

By this time, hopefully, we have come to terms with how JAXB can be used in a J2EE-based enterprise application. Even in a simple example like the one described in this article, where can we see this being used? Say, for example, there is a book search capability that allows nearby libraries to search for a list of books in CityLibrary that matches a specific criterion (for example, books by a certain author). The results of the search can be easily returned in an XML format using code that is similar to the marshallIt method in our example. In more sophisticated examples, a J2EE application can accept XML documents (conforming to the XML Schema from which Java artifacts have been generated a priori) and then use JAXB to create Java objects that can then be easily used in order to perform application-specific processing.

An application's domain object model is an ideal candidate to be modeled in an XML Schema. Using JAXB, the transformation between the XML representation and its corresponding Java object tree can be simplified. This will allow XML information exchange between various application tiers and even between communicating applications – a very simple piece of work.

Conclusion

This two-part series attempted to expose you to the world of JAXB. The conceptual introduction in the first part is concluded in the second part with an illustrative example of the usage of JAXB in a J2EE-based application.

The bridge between XML and Java has really been narrowed with the advent of JAXB, and I hope that this article series helps to illustrate this to the *JDJ* reader community. ☺

Resources

- *JAXB User's Guide*: <http://java.sun.com/xml/jaxb/users-guide/jaxb-using.html>
- *JAXB Specification*: <http://java.sun.com/xml/downloads/jaxb.html>

- *JAXB API Specification*: <http://java.sun.com/webservices/docs/1.3/api/index.html>
- *JAXB Reference Implementation*: <http://java.sun.com/webservices/downloads/webservicespack.html> (Note: The Reference Implementation of JAXB comes packaged inside the Java Web Services Developer's Pack [JWS DP]. Once this is installed, the JAXB compile time and runtime libraries are available in the <install-root>\jaxb directory).

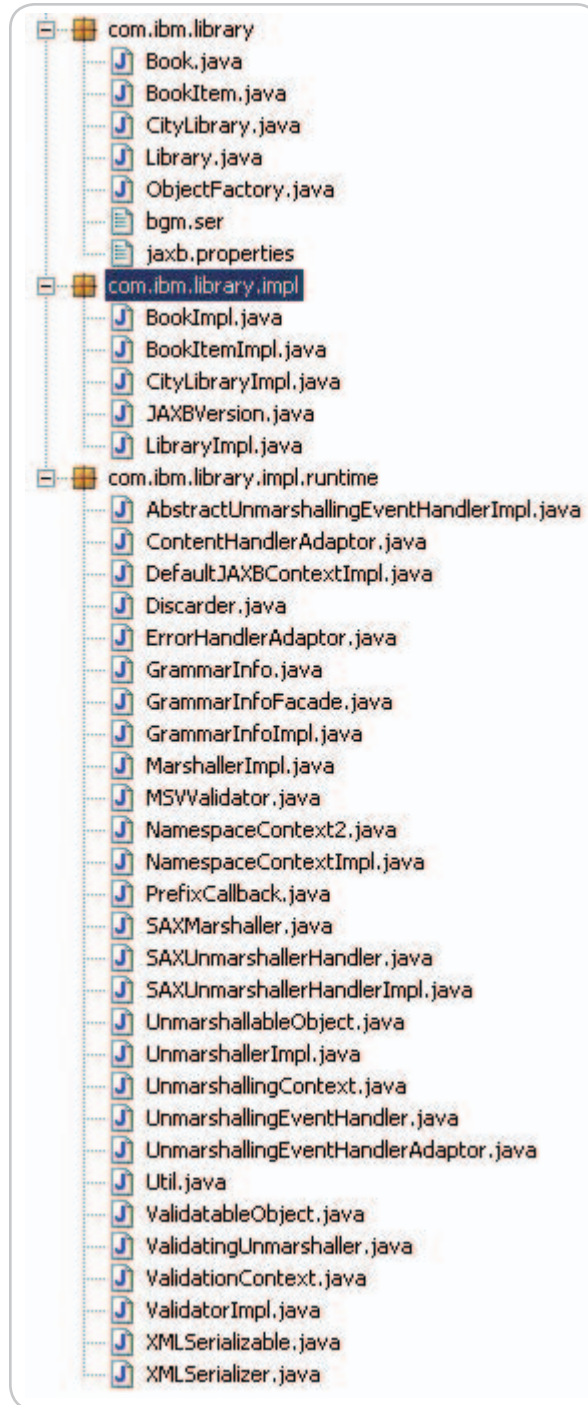


Figure 1 List of Java artifacts produced by the JAXB compiler



DESKTOP



CORE



ENTERPRISE



HOME

Listing 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
  jxb:version="1.0"
  xmlns:domainObjects="http://www.ibm.com/domainobjects"
  targetNamespace="http://www.ibm.com/domainobjects"
  xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
  jxb:extensionBindingPrefixes="xjc">

<!-- Global Settings -->

<xsd:annotation>
<xsd:appinfo>
  <!-- JAXB Global bindings initiation reference:

  <globalBindings>
  [ collectionType = "collection" ]
  [ fixedAttributeAsConstantProperty= "true" | "false" | "1" |
"0" ]
  [ generateIsSetMethod= "true" | "false" | "1" | "0" ]
  [ enableFailFastCheck = "true" | "false" | "1" | "0" ]
  [ choiceContentProperty = "true" | "false" | "1" | "0" ]
  [ underscoreBinding = "asWordSeparator" | "asCharInWord" ]
  [ typesafeEnumBase = "typesafeEnumBase" ]
  [ typesafeEnumMemberName = "generateName" | "generateError" ]
  [ enableJavaNamingConventions = "true" | "false" | "1" | "0"
]
  [ bindingStyle = "elementBinding" | "modelGroupBinding" ]
  [ <javaType> ... </javaType> ]*
</globalBindings>
-->

<jxb:globalBindings collectionType="java.util.ArrayList"
  fixedAttributeAsConstantProperty="true"
  generateIsSetMethod="false"
  enableFailFastCheck="false"
  choiceContentProperty="false"
  underscoreBinding="asWordSeparator"
  typesafeEnumBase="xsd:NCName"
  typesafeEnumMemberName="generateError"
  enableJavaNamingConventions="true"
  bindingStyle="elementBinding">

<xjc:serializable />

</jxb:globalBindings>

<!-- JAXB schema bindings definition reference:

<schemaBindings>
[ <package> package </package> ]
[ <nameXmlTransform> ... </nameXmlTransform> ]*
</schemaBindings>

where:

<package [ name = "packageName" ]
[ <javadoc> ... </javadoc> ]
</package>

and:

<nameXmlTransform>
[ <typeName [ suffix="suffix" ]
[ prefix="prefix" ] /> ]
[ <elementName [ suffix="suffix" ]
[ prefix="prefix" ] /> ]
[ <modelGroupName [ suffix="suffix" ]
[ prefix="prefix" ] /> ]
[ <anonymousTypeName [ suffix="suffix" ]
[ prefix="prefix" ] /> ]
</nameXmlTransform>
```

```
-->
<jxb:schemaBindings>
  <jxb:package name="com.ibm.library" />
</jxb:schemaBindings>

</xsd:appinfo>
</xsd:annotation>

<xsd:element name="CityLibrary" type="domainObjects:Library" />
<xsd:element name="BookItem" type="domainObjects:Book" />

<xsd:complexType name="Book">
  <xsd:attribute name="title" type="xsd:string"/>
  <xsd:attribute name="author" type="xsd:string"/>
  <xsd:attribute name="isbn" type="xsd:integer"/>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:annotation>
      <xsd:appinfo>
        <jxb:property name="Books" />
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:element ref="domainObjects:BookItem" />
  </xsd:choice>
</xsd:complexType>
</xsd:schema>
```

Listing 2: Output of the JAXB compiler

```
parsing a schema...
compiling a schema...
com\ibm\library\Book.java
com\ibm\library\BookItem.java
com\ibm\library\CityLibrary.java
com\ibm\library\Library.java
com\ibm\library\ObjectFactory.java
com\ibm\library\bgm.ser
com\ibm\library\jaxb.properties
com\ibm\library\impl\runtime\ValidatableObject.java
com\ibm\library\impl\runtime\ValidatingUnmarshaller.java
com\ibm\library\impl\runtime\PrefixCallback.java
com\ibm\library\impl\runtime\GrammarInfoFacade.java
com\ibm\library\impl\runtime\Discarder.java
com\ibm\library\impl\runtime\UnmarshallingEventHandlerAdaptor.java
com\ibm\library\impl\runtime\SAXUnmarshaller.java
com\ibm\library\impl\runtime\NamespaceContextImpl.java
com\ibm\library\impl\runtime\ErrorHandlerAdaptor.java
com\ibm\library\impl\runtime\UnmarshallerImpl.java
com\ibm\library\impl\runtime\SAXUnmarshallerHandler.java
com\ibm\library\impl\runtime\MSVValidator.java
com\ibm\library\impl\runtime\UnmarshallingContext.java
com\ibm\library\impl\runtime\XMLSerializer.java
com\ibm\library\impl\runtime\AbstractUnmarshallingEventHandlerImpl.java
com\ibm\library\impl\runtime\XMLSerializable.java
com\ibm\library\impl\runtime\UnmarshallableObject.java
com\ibm\library\impl\runtime\GrammarInfo.java
com\ibm\library\impl\runtime\NamespaceContext2.java
com\ibm\library\impl\runtime\MarshallerImpl.java
com\ibm\library\impl\runtime\ContentHandlerAdaptor.java
com\ibm\library\impl\runtime\Util.java
com\ibm\library\impl\runtime\DefaultJAXBContextImpl.java
com\ibm\library\impl\runtime\ValidationContext.java
com\ibm\library\impl\runtime\ValidatorImpl.java
com\ibm\library\impl\runtime\UnmarshallingEventHandler.java
com\ibm\library\impl\runtime\GrammarInfoImpl.java
com\ibm\library\impl\runtime\SAXUnmarshallerHandlerImpl.java
com\ibm\library\impl\BookImpl.java
com\ibm\library\impl\BookItemImpl.java
com\ibm\library\impl\CityLibraryImpl.java
com\ibm\library\impl\JAXBVersion.java
com\ibm\library\impl\LibraryImpl.java
```

Achieve higher quality in less time, with fewer resources.



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

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

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

Automated testing and standards compliance for any team configuration.

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

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

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



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

Availability

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

Contact Parasoft for details and pricing information.

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

Moving to a Cluster...

What the guidebook doesn't tell you

by David Purcell

You've engineered a J2EE application that has become mission critical for your business operations. You know that downtime will be less acceptable as the business starts to rely more on the application, so you want to start eliminating single points of failure and improve availability. One of your first thoughts might be: "Let's move to a clustered application server environment."

Migrating to a clustered environment is a reasonable thing to do, but if this is your first experience with clustering application servers, make sure you understand what you're getting into when you go to your project owner and tell him or her about your plans.

Project Considerations

Why Are You Migrating to a Cluster?

Before anything, make sure you're migrating to a cluster at the right time and for the right reasons. A clustered application server environment can give you failover capabilities, which will help you sleep a little better at night and should help distribute the load a little bit, delaying the point at which your hardware becomes inadequate. If you're squeezed by tight budgets, make sure you're tackling the highest priority issue in your system so you make the most of your limited resources.

What are the consequences of not doing it? Business owners who have lean budgets might need to understand the chances of downtime occurring and its implications. Also, you could consider alternatives, such as a multiserver environment that doesn't involve clustering. Clustering can give you advantages such as:

- Load distribution
- High availability
- Session failover
- EJB sharing
- Centralized control

However, there are other ways to distribute the load among servers, and you might not be using EJBs for persistence, so you wouldn't need to take advantage of EJB sharing. Adding other server instances in a nonclustered fashion could give you the high availability you desire. Your decision to go with a cluster might come down to whether

or not it is important that all users who had been hitting a server keep their session intact and not notice if that server goes down. The bottom line is that there is a cost-benefit consideration that needs to be made, and, usually, if there are external customers or important business processes involved, it is easy to see how failover and clustering can be important. Just remember that you have options.

In any case, be sure that there is a need and be ready to discuss not only the migration costs, but also some of the ongoing costs described below.

One Change at a Time, Please

Migrating to a cluster isn't always straightforward. Don't make your life more difficult by changing the application's functionality as part of your project. If you are managing a migration project, make sure that business owners don't try to tack on new functionality or other application changes as part of the project. After rollout, you'll need to be able to determine if a problem is related to the new environment. Adding new functionality will make troubleshooting that much more difficult after rollout. Also, you'll want to be able to make a fair comparison between the old and new environments based on metrics and measurements, identifying if performance has improved after the migration; changes in functionality will skew such measurements.

Test the Failover Capabilities

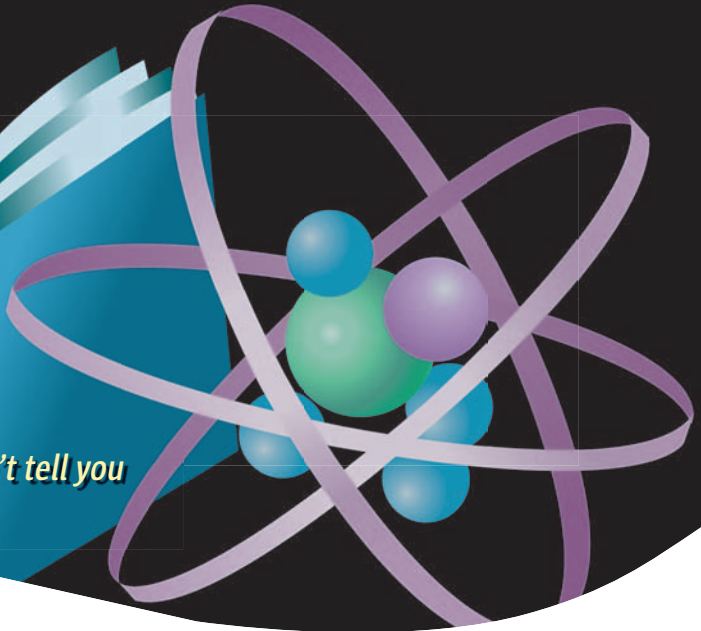
Part of a project plan to deploy the cluster should be to shut down servers in the cluster and see how well the remaining applications perform. Besides verifying the basic failover capabilities, you might uncover some code design points that do not support the cluster.

Be sure to test the system by shutting down each application server node (leaving at least one node up), and also by shutting down individual applications within the node (leaving at least one instance of the application running), and verifying that each combination still results in a working system.

Administration Considerations

Cluster Other Environments, Not Just Production

You might be thinking that since it is the same application, why bother clustering the other environments, such as the



DESKTOP



CORE



ENTERPRISE



HOME



David Purcell is a Web applications supervisor for the Minnesota State Colleges and Universities system.

He has been a senior applications architect in the IT industry for nearly a decade, providing technical leadership on Web application projects of all sizes.

david@supercell.org

certification or staging environments? They don't need the failover capabilities. Think again. First, you'll need to make sure that you have set up the environments and configured the servers properly for the cluster. What better way to start than in an environment that won't become a production environment? More important, however, is that to truly test your application, you'll want your test environment to look just like your production environment. If a problem exists with your setup, you need to find it.

Setting up the certification environment to match the production environment is standard practice for many organizations. However, the extra costs of hardware and server licenses might make departments on a lean budget try to get by without setting up identical environments. In the end, the costs of not doing so will probably catch up with them. On a recent project, for instance, I saw how a machine hosting a directory server masked an underlying problem in the certification environment, as its hardware was different than the production environment's. Although the problem existed in both environments, it manifested itself differently in the certification environment. The problem wasn't noticed until the system was deployed into the production environment, and the trouble that it caused resulted in downtime and significant troubleshooting expenses. Having identical environments will save you time and money in the long run.

Little Things Add Up

Be prepared for a brief period of frustrations while getting to know the new environment. Your cluster's behavior might have some "nuances" that take some getting used to. For example, in a recent migration, we saw an unusual problem appear for the first time as we deployed an application to production. In our case, the two servers in the cluster were producing different results. This wasn't supposed to happen, and we were able to resolve it, but now we know what to do in that unusual circumstance. Little tricks like that are learned only from experience. Expect some initial rough patches.

Also, debugging might become a little more tedious. For example, someone may experience a problem, but you don't know which instance of the server handled the request. When trying to debug an issue in a nonproduction environment, you might want to shut down one instance of the server so you can more quickly focus on the problem.

Don't Let the Redundancy Fool You

Just because you have redundancy built into the system, don't let that fool you into thinking you can be a little more lax with respect to administration practices. Don't take down an instance of the cluster during the day to perform maintenance if you wouldn't have done that before you had the cluster. Of course, large systems with many instances in the cluster are designed to do just that, but small operations need to make sure they don't abuse their new clustering capability. The bottom line is, you probably still need to perform maintenance tasks after hours.

Application Design Considerations

Primary Key Generation

There are many techniques for primary key generation, but not all are cluster-safe. Be sure you understand how your primary key generation works and see if it should be used in a cluster.

Persistence mechanisms, such as Hibernate, should identify which of their primary key generators are safe when used in a cluster. A mechanism we used had a seemingly simple CMP bean implementation of a primary key. Although it appeared to be cluster-safe, the entity bean that kept track of the latest primary key only wrote to the database for every 10 requests it received. Therefore, in a situation with multiple servers, each one eventually got to the tenth item and wrote to the db, overwriting the previous value.

Not Everything Can or Should Take Advantage of Clustering

In a clustered environment, most application servers will share some objects, so it doesn't matter which instance of the application server handles a request. However, some objects are not going to be able to take advantage of clustering. File services or timed tasks, for example, will not take advantage of being shared between instances of the application server, and probably shouldn't.

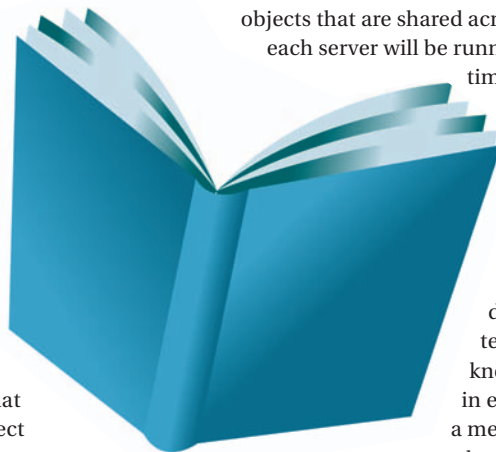
If you have objects that use timed tasks, such as classes that implement `java.util.TimerTask`, those tasks will not be shared across the cluster, even if they are associated with objects that are shared across the cluster. The result is that each server will be running its own tasks. So if you have

timed tasks and you don't want multiple instances of the task to be running, you need to come up with a way around it. Clark Richey's article, "Clustered Timers" (*JDJ*, Vol. 9, issue 3, www.sys-con.com/story/?storyid=43944), provides a good discussion on timers in a clustered environment. In our case, we knew that the timers were running in each server, but we implemented a mechanism where they had to check a database record to see if they should continue with their process. Only one instance of the timer would be allowed to perform its process.

Watch Out for File Services

File services also need to be managed carefully if you have a clustered environment. If you have a process that is writing or reading files from a file server, you need to make sure that each instance of the application server can access that same file server location. That means you need to have identical mount points established on each machine. For Windows machines, it's best to use UNC (universal naming convention) when defining a location on a file server, rather than a Windows mapped drive, as the mapped drive only exists when a user is logged on to the machine, which means that it might not always be available.

If you have a situation where you have lots of writes to only one file, you might want to consider having each application server maintain its own instance of the file, and



then setting up a nightly cron job to merge the files. If you don't want to mess with that kind of maintenance issue, or the merge won't meet your business needs, consider changing that type of function to use a database instead of writing to a file.

JSP/Servlet Clustering and HttpSession State Failover

One of the primary features of clustered application servers is the ability to cluster servlets and JSPs by sharing HttpSession objects across the cluster. The advantage of this is that if one server in the cluster goes down, the user's HttpSession remains, and the user doesn't notice the difference.

I've already mentioned how some objects can't be clustered, so those objects are certainly ones that you wouldn't want to put into the HttpSession. However, keep in mind that since the HttpSession objects are going to be kept on other servers, the memory requirements on each server might not be reduced much by distributing the load. Usually the HttpSession will be kept in memory on the server that the user first visits, and another server hosts a replica of the HttpSession. If you only have two servers, each server will hold the entire set of all HttpSession state objects, not half of them.

When designing applications for a clustered environment, keep in mind that your session objects should be serializable, you should keep the session objects small, and you shouldn't overuse the HttpSession by keeping everything in it. When your project is being tested, make sure you test the failover capabilities of the servers to verify that the HttpSession is always available and users don't notice when a server is taken offline.

EJB Handles and Distributed Applications

When you had an application deployed in a nonclustered environment, you might have been able to take advantage of knowing where your applications lived. For instance, in a distributed environment, if one application had to call an EJB on another server, the calling application could cache the EJBHome handles in a ServiceLocator, rather than performing a lookup each time the handle was needed. This would give you a slight performance improvement. However, if the EJBs were now distributed among a cluster, that same cache would result in a situation in which an application instance would always call the same server to get the EJB, rather than distribute the requests among the different instances. The caching would result in nullifying some of the advantages of the cluster. To complicate the matter, the application would appear to work fine, so you might not be able to see the problem until one instance of your cluster had to shut down.

EJB containers on many of the major application servers provide mechanisms to handle distributing the load among clusters. BEA's WebLogic server, for instance, allows you to specify in the deployment descriptors that an EJB is clusterable. In that event, the EJBHome stub, and possibly the EJBObject remote stub, are aware of the cluster (replica-aware stubs), and will try to find a different server if the initial call fails. Whether or not an EJB can use a replica-aware EJBObject remote stub depends on the type of EJB and, for entity EJBs, the concurrency strategy (read only or read-write) selected at deployment time.

IBM's WebSphere provides a concept called EJB Workload Management. No specific deployment configuration changes are required for the EJBs to be clusterable, as long as the EJB client makes requests through the WLM Plug-In in the client application server.

WebSphere supports clustering of stateless session beans and the clustering of stateful session bean home objects among multiple application servers. However, it does provide clustering of a specific instance of a stateful session bean. Each instance of a specific stateful session bean can exist in only one application server, so once the bean instance is created, requests to the bean must be directed to that particular application server. For entity beans, the WebSphere EJB container supports three different entity bean caching options that define where the bean can be accessed (which servers) and when it is reloaded and passivated.

Be sure to take the time to understand the clustering capabilities of your container with respect to EJBs. If such clustering capabilities aren't available, don't cache the EJBHome handles in your client. Again, testing the failover capabilities of your server is important. You will want to make the changes that you think are needed, and then test failover to make sure there aren't any dependencies between servers.

Conclusion

Migrating from a single-server environment to a clustered environment may sound straightforward, but you need to enter such an endeavor with the appropriate expectations. Certainly, you need to understand the technical implications of using a cluster, including the configuration, administration, and software design changes that you need to make. However, you also need to make sure the migration project team has the right expectations, takes the time to test the clustering capabilities thoroughly, and doesn't convolute the project with changes to your application. If you prepare yourself with some of these steps, your migration effort will be much smoother. ☺



Migrating to a cluster isn't always straightforward.
Don't make your life more difficult by changing the application's
functionality as part of your project”

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

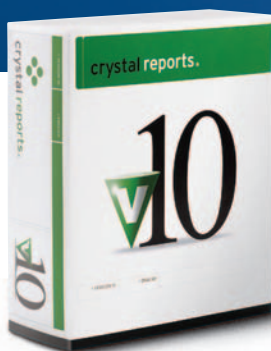
Crystal Reports 10

Which edition of Crystal Reports® is right for you?

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

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

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



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

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



Using JNDI...

... to build flexible, technology-independent enterprise systems

by Rost Vashevnik

A challenge of software architecture is to create software that can grow with the business and withstand changes to the technology with minimal redevelopment costs.

Business growth usually means increased loads on enterprise computer systems. As more customers and staff come on board, they put more transactions through, and may also require more complex security and access control requirements. To meet these demands, the well-architected enterprise system should be able to morph and scale as much as possible without a major redevelopment effort. Therefore it seems a good idea to design flexible and scalable multitier enterprise systems from the start. However, at the beginning, projects are often neither able nor willing to invest in the extra complexities required to implement such systems. All too often enterprise systems start life as a simple two-tier JSP + database solution and are later reengineered or completely rewritten as a three-tier or even four-tier architecture at great additional cost.

This article describes a simple approach to enterprise systems design and shows how the simple customization of the Java Naming and Directory Interface (JNDI) mechanism can be used to build highly adaptable and flexible applications where system features such as logging, caching, distribution, transaction management, asynchronous or synchronous invocations, and access control are bound into the application after it was built without the need to reengineer or even rebuild the application code.

The approach described in this article is an integral part of the MetaBoss Target Software Model and it has been used very successfully by the users of the MetaBoss. MetaBoss is an

integrated suite of tools for the design, development, and management of software systems through modeling. It utilizes Model Driven Architecture concepts and is primarily oriented toward enterprises using Java-based tools and technologies. MetaBoss is a part of the growing family of Professional Open Source products. It is dual licensed and can be used under the Open Source GPL license or MetaBoss Commercial license. More details are available from www.metaboss.com.

Danger of Mixing Technology and Business Code Together

There are many fine enterprise technologies out there and this article does not discuss the merits of using one rather than another. Almost all of them,

however, have one dangerous feature – if they're used in the business code layer without extra precautions, they tend to impact the application code to a point where the application becomes inflexible from the technological refactoring point of view. The business application developer who is coding to the standard recommended (or even enforced) by any one of these platforms will likely mix the code dealing with technology-specific issues with pure business logic code. The resulting software is inflexible and too strongly coupled with the chosen technology.

This impact can be observed in practically all areas where technology touches the application code. Here are some examples offered by remote invocation technologies.

A Few Words About Aspect-Oriented Programming

The last few years have seen the emergence of a new paradigm called Aspect-Oriented Programming (AOP). This mechanism allows the “injecting” of new behavioral features into certain points of existing application classes. This approach allows the separation of secondary functionality that the main implementation code shouldn't be concerned about.

AOP is a great approach and it can be used to separate the business code from technology. However, my experience is that it has a few weak points.

First, the AOP paradigm is not native to the Java language. Most of the AOP frameworks include the need to have an XML document or Javadoc tags that define the join points in the main code and the need to post process the Java byte code in order to “implant” the callbacks from the main code to the secondary code. Some other frameworks take a different approach and extend the Java language, which adds quite a bit of complexity and requires a special compiler. The bottom line is that AOP is not native to Java.

Another important weakness is that the main body of code has no idea or control over where join points will be located. Most of the AOP frameworks allow you to place join points almost anywhere without any limitations, such as entry to or exit from any method or access to any variable. This approach can present a problem if unsuspecting main code is impacted by something occurring in the advice code or visa versa. Examples of this are multithreading or locking, long execution time, and unexpected exceptions.

To illustrate why this lack of control may be dangerous, imagine my car's owner's manual. When talking about changing flat tires, it says “Be sure to use designated jacking positions provided on the car.” When talking about towing it says “Vehicles fitted with IRS (Independent Rear Suspension) should always be tray towed.” The manufacturer of my car has provided certain, well-defined join points for the lifting device and has not provided join points for tow cables simply because this particular car cannot be pulled. No doubt, attempts to ignore these original design limitations will be very damaging to my car.



Rost Vashevnik

is a principal architect of MetaBoss, an open source MDA tool suite. He works as a software architect consultant in Australia.

rost@metaboss.com

The typical requirement is to use prescribed super interfaces to represent remote objects. Most of the technologies require that Java classes and interfaces representing remote objects implement certain interfaces or extend particular abstract classes. In CORBA we must use org.omg.CORBA.Object as a super interface of the remote service object. In J2EE we must use various super interfaces to build enterprise beans. The Web service interface too (at least when using JAX-RPC) requires you to extend java.rmi.Remote. I do acknowledge that most of these prescribed interfaces are very simple to implement, and typically there is absolutely nothing to do except specify that the class implements the interface. However, a side effect is that a number of technology-specific classes, interfaces, and their methods are visible to the business programmer.

The typical requirement is to use prescribed value types to represent remote call parameters. Most of the technologies document a list of supported Java value types that can be used as parameters passed in and out of the remote services. These lists are

typically very rich and most of the commonly used types are supported. Each technology, however, still limits the number of Java types that can be used "as is" and leaves it up to the application code to deal with the others. This leads to technology limitations creeping up into the remote services interface design.

The typical requirement is to catch and process special exception types. Most of the technologies communicate network failures to the application layer via special types of exceptions specific to the particular technology. This means the application code may need to catch these technology-specific exceptions.

The typical requirement is to use prescribed coding patterns. Most of the technologies require the use of very specific coding patterns, especially when it comes to connecting to or disconnecting from the remote service, using pervasive services, etc. For example, to make a remote call to the J2EE enterprise bean, the client needs to obtain an instance of the bean home interface via JNDI, then use it to obtain the instance of the bean remote interface and finally make the

remote call. For a CORBA client to do the same remote call, it has to use the ORB singleton to connect to the naming service, then use the naming service to obtain a remote object reference, then use the Helper class to narrow the reference, and finally make a call.

To illustrate what may happen when these technologies are not insulated from the application code, consider the following examples.

An application programmer of a CORBA-based system has decided to use an array of org.omg.CORBA.Object elements to keep or pass around the list of previously called business services. This somewhat short-sighted decision was made because org.omg.CORBA.Object was a very convenient common super interface for all of the services. This decision entrenches CORBA technology into the business code and makes it more difficult to move this implementation to any other technology.

An application programmer of a J2EE-based system needs to create an entity bean dealing with value types from the java.awt.geom package. The types in this package are value ob-



FIND SOMETHING BETTER.



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

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

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

DiceTM
Look to the tech leader first.TM

©2004 Dice Inc.

jects that help to represent geometric shapes such as Arc, Rectangle, etc. However, for a reason unknown to us they don't implement a Serializable interface (at least as of JDK 1.4.2). This means the application programmer will probably have to make the entity bean accept individual attributes comprising these values (i.e., separate X, Y, Height, and Width attributes instead of a Rectangle2D instance). The net effect is that the business application component design is impacted by the deficiencies of the technology.

An application programmer of a simple single-tier system has not given any attention to the future scalability and distribution requirements and has created a simple application using plain Java classes, without any logical layering or split of responsibilities between them. This decision makes it harder to split the system into the separate distributed components at a later stage.

Looking for a Better Approach

When we started to devise the MetaBoss Target Programming Model and adopt coding patterns for our code generators to conform to, we decided to aim for the ideal business "friendly" programming model. Such a model must be as technology-independent as possible, to the point where application code looks exactly the same irrespective of how the system will be deployed, and the actual deployment topography and technological mechanisms can be chosen

after the application code is built. At the same time we wanted to stay within the standard facilities offered by the Java language as it exists today.

To achieve these goals we designed the programming model based on technology-independent business components and the use of the JNDI mechanism for application assembly (i.e., connecting components with their clients).

JNDI is an abstract framework that provides naming and directory functionality for Java programs. We chose JNDI because it has a number of advantages:

- The mechanism is native to Java because the JNDI framework is contained inside J2SE and therefore available in any JVM. The Java community is familiar with JNDI usage patterns, and the component lookup pattern, which is used the most, is relatively easy to learn.
- It offers complete separation of interface and implementation. The client has to only be aware of the component interface and some kind of location string for use in the lookup operation. The job of finding or creating the instance of the interface is delegated to the JNDI framework and the underlying naming and directory mechanisms.

The Basics of the MetaBoss Target Programming Model

From the component user point of view, the programming model has the following features.

Application code is split into components. Each component has a certain well-defined set of responsibilities. This division is driven entirely by the business logic, business needs, and logical layering of the application. Most important, it's not limited or dictated by technology issues. As such, the decision to have an entity-like component or service-like component can be made in any application, not necessarily a J2EE one. This means the component boundaries are the only places where platform mechanisms, such as remote invocation or caching, can be plugged in.

Each component is exposed to its clients via the component interface – the plain Java interface that's not required to have any properties beyond those required by business logic. Methods in these interfaces are able to use any Java class as a parameter and are able to declare and throw any number of any kind of exceptions. As before, there are no technology-motivated requirements to include anything in the input or output parameter list or to throw any particular kind of exception.

Each component interface is identified by the unique identifier – we call it the component URL. The component URL is a string that's formed as follows:

```
component://<fully qualified name of component interface>
```

Each component interface exposes a string constant named COMPONENT_URL that contains the unique identifier of the interface. Listing 1 contains the sample of the typical component interface with COMPONENT_URL string constant. Apart from the prefix, COMPONENT_URL is simply the fully qualified name of the Java interface exposed by this component.

To obtain an instance of the component, client code has to do a simple JNDI lookup (see Listing 2, lines 10 and 11). Note, the client code only needs to import and be aware of the component interface and nothing else. The actual instantiation or lookup of the particular component implementation occurs at runtime.

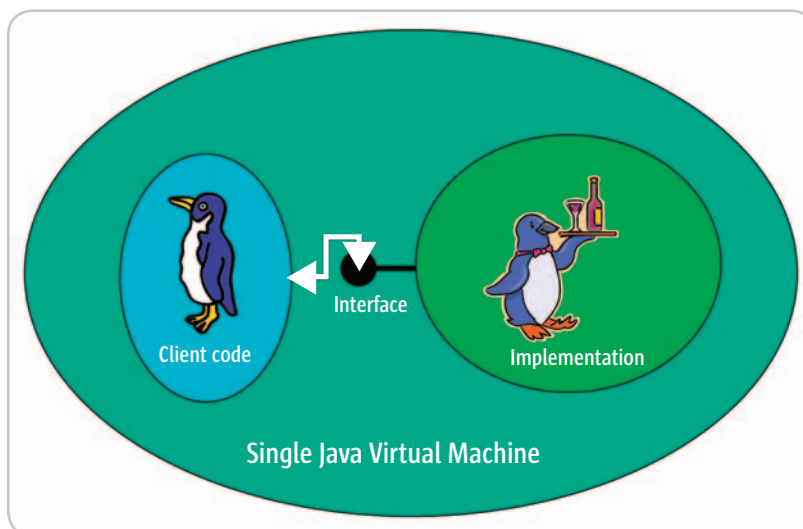


Figure 1 Simple configuration

It may appear that this pattern looks similar to J2EE. It does; however, there are several key differences. Meta-Boss components expose pure business logic interfaces, which are not polluted by technology-related “small print” as occurs with enterprise beans. Moreover, the designer of these components is not concerned about which remote invocation mechanism, if any, will be used. This means application components promise to perform logical operations without disclosing where the operation will be executed and how request and response signals will be transmitted (if indeed such a need to transmit exists).

From the component implementer point of view, the programming model has the following features:

- Each component can have an unlimited number of different implementations. Implementations may be of the “proxy” kind or the “real implementation” kind. The proxy implements some secondary feature and calls another implementation of the same component to do the actual work. The real implementation fulfills the main business task of this component.
- Each component implementation consists of two classes: the component implementation class and the implementation factory class.
- The component implementation class must implement either the component or the `java.lang.reflect.InvocationHandler` interface:
 - **Component interface:** This approach produces the “strongly typed” dedicated implementation, which can only be used as the implementation of this particular component type. It’s more typically used for the real implementations. Listing 3 shows an example of this kind of implementation.
 - **Generic `java.lang.reflect.InvocationHandler` interface:** This approach produces the “loosely typed” component implementation, which may be used as the implementation for many different component types. It’s more often used to produce reusable proxy implementations. Listing 4 shows an example of the simple logging proxy implementation. (Listings 4–6 can be downloaded from www.sys-con.com/java/sourcec.cfm.)
- The implementation factory class is a simple class that must implement the JNDI standard `javax.naming.spi.ObjectFactory` interface. The factory is only required to implement a single `getObjectInstance()` method that should return a new or cached component implementation. Listing 5 shows a simple factory that returns new implementation instances every time. More complex factories may cache implementation objects and return them multiple times.

A Few Practical Examples

Having created an application that follows this programming model, what can now be done with it, and how can the promised flexibility be used in practice?

Figure 1 shows a simple way to deploy our application where the component client and the component implementation code run in the same JVM without any additional mechanisms plugged in. It works well for the initial deployment of a simple application, perhaps a basic JSP

application deployed entirely in a Java servlet engine such as Tomcat. Our experience has shown that this configuration is favored by business application developers because it offers them a way to test the business logic without the complexities of a full distributed deployment. In this scenario, JNDI is configured to return the actual component implementation as a result of the component lookup operation.


Figure 2 shows the introduction of the “invisible” architectural feature through the use of a special proxy. It works well for pluggable security, logging, or caching mechanisms. JNDI is configured to return an instance of the special proxy instead of the actual component implementation. Once the special proxy is invoked, it can do anything it likes before and/or after invoking the actual underlying implementation. The process of invoking the underlying implementation from the special proxy implementation is also based on JNDI lookup. Thanks to this use of the JNDI lookup pattern inside proxies, chains of proxies can be built. This means that the proxies created are simple, single purpose classes.

Figure 3 shows the introduction of the “invisible” remote invocation mechanism, again through use of a special proxy. In this case the special proxy consists of two parts: client and server. At the client side JNDI is configured to return the instance of a remote proxy client instead of the actual component implementation. This proxy client makes remote calls to the proxy server, which in turn obtains the underlying actual implementation, again via

Google Seeks Expert Computer Scientists

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

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



JNDI. Similar to the previous example, this offers an opportunity to build chains of proxies. As an example, the logging proxy could be configured to run on the client side, server side, or even both sides of the remote invocation proxy.

A key point to notice is that in all the above examples the original component implementation code and the component client code were not modified or rebuilt in any way.

Under the Hood

The smarts to this approach are hidden inside the JNDI provider mechanism. JNDI is a well-supported standard and many application servers provide a Java objects repository facility with a JNDI interface, allowing it to store and retrieve many different types of Java objects. These repositories, however, are not quite what is needed for this mechanism to work. The unique feature required is the ability to serve different kinds of implementations of the same component to different callers. This facility is fundamental for proxy chaining, where lookup from the client must return a proxy implementation, but lookup from the proxy must return the “real” one or the proxy that is next in the chain.

This is why MetaBoss includes a special JNDI service provider implementation. This implementation is packaged in a single `MetaBossComponentNamingProvider` Java archive, available from www.metaboss.com. It can be used as a standalone library, totally separate from the rest of the MetaBoss suite. It has the following features:

- A URL context factory implementa-

tion that looks after the component scheme (i.e., all URLs starting with component: prefix).

- It only supports lookup operations. Most important, it doesn't support the bind operation. During lookup, after the decision to return a certain implementation is made, the JNDI Object Factory corresponding to the chosen implementation type is loaded and invoked “on the fly” in order to obtain the instance of this implementation.
- It uses the set of client/interface/implementation mapping properties to understand which implementation needs to be returned from the particular lookup operation.
- It can search the set of directories for JAR files with required implementation classes and dynamically load them into the isolated class loaders.
- As with every JNDI service provider, it is in itself a plug-in, which can easily be replaced without any impact on the application code. For example, one of our clients has replaced this implementation with the one that gets mapping instructions from the database instead of system properties.
- As with most other JNDI service providers, to plug it in you only need to put it on the main application class path.

For interested readers who wish to read more, I suggest downloading the source from www.metaboss.com and taking a look at it. I also recommend studying the basics of how to build a JNDI service provider before delving into the source.

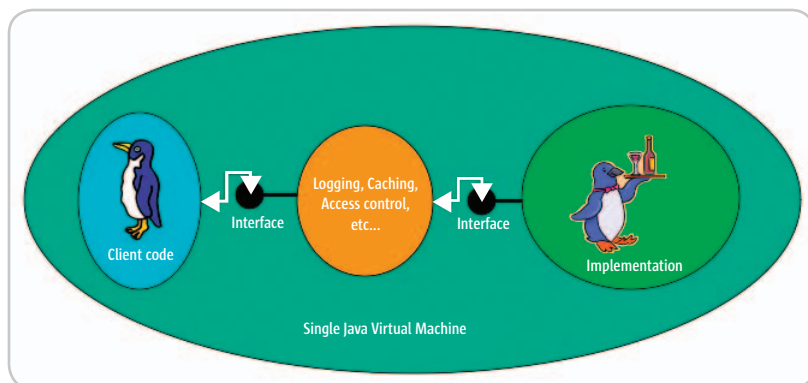


Figure 2 “Invisible” architectural mechanism

The Last Piece of the Puzzle – The Implementation Mapping

This article has shown how to develop dedicated implementations of our components as well as generic ones. Thus far we have not really done much over and above what is required when building a well-architected solution. Now all that is left (apart from downloading the MetaBoss Component Naming JNDI Service Provider plug in) is to configure the mapping rules defining “who gets what,” in other words what kind of implementation or chain of implementations has to be served when the particular client or family of clients is looking up the particular interface. The “out of the box” implementation supports reading these mapping rules as a set of provider-specific JNDI environment properties. JNDI allows for these properties to come from a number of locations. However, in our experience we tend to favor “out of code” locations such as application resource files.

The mapping entry is a key=value pair where the key describes the lookup operation (in terms of who is looking up what) in the form:

```
com.metaboss.naming.component.<interface
match expression>[</client match expres-
sion>]
```

and the value describes what has to be returned from the lookup in the form:

```
<implementation match expression>[(<implem-
entation match expression >{...})]
```

In more detail:

- **com.metaboss.naming.component:** A constant prefix used to distinguish particular provider-specific properties.
- **Interface match expression:** A mandatory expression used to identify single or multiple component interfaces. It may contain wildcard characters.
- **Client match expression:** An optional expression identifying the place in the code from which the interface is being looked up. This place in the code can be identi-

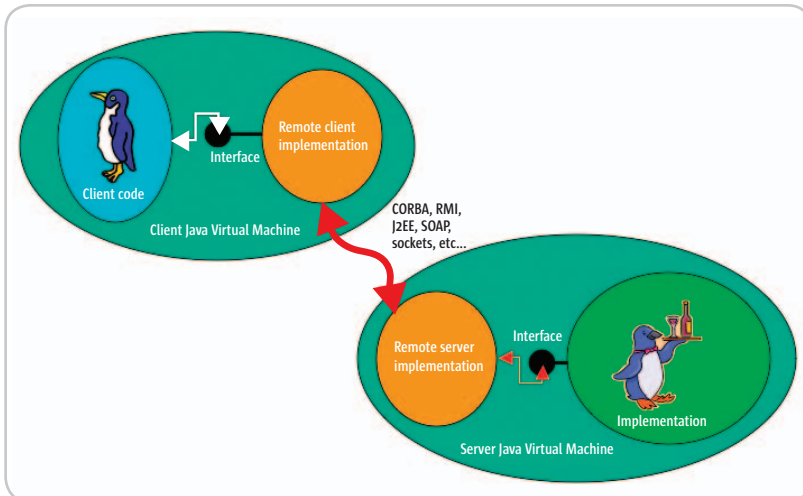


Figure 3 “Invisible” distributed deployment

fied with any precision up to and including the name of the method from which the lookup has come. It can contain wildcard characters. The mapping entry where the client match expression is not specified is used for all clients not individually configured.

- **Implementation match expression:** A mandatory expression identifying the object factory to be invoked in order to obtain an implementation. As shown above, these expressions can be chained in order to define the chain of

implementations. Wildcards are not allowed since one and only one implementation must match. However, a small number of predefined keywords makes it possible to reference parts of the interface and/or client names in this expression.

Listing 6 shows a few sample mapping entries. As you can see, the mapping syntax offers considerable flexibility. The matching of the particular lookup operation tries the more explicit mapping entries (i.e.,

entries where the key is less vague) before attempting the less explicit ones.

Summary

The mechanism described here offers an architecturally neutral, component-oriented approach to writing the business applications. It protects software development investment by separating business and architectural concerns. While providing many features found in Aspect Oriented Programming frameworks, it stays within the boundaries of the core JDK. I have found this mechanism to be very useful and successful on a number of complex enterprise projects. It’s available from www.metaboss.com and can be used with or without the rest of the MetaBoss suite. ☺

References

- *Raw JNDI knowledge:* <http://java.sun.com/products/jndi/tutorial>
- *MetaBoss Component framework:* www.metaboss.com
- Burke, B. “It’s the Aspects: A new paradigm.” *JDJ*, Vol. 8, issue 12.
- *AspectJ, the Aspect Oriented Programming framework, part of Eclipse project:* <http://eclipse.org/aspectj/>

Listing 1

```
01 package com.metaboss.sdlctools.services.codegeneration;
02
03
04 /** This component offers operations dealing with code generation. */
05 public interface BSCodeGenerator
06 {
07     /** Naming URL of the component */
08     public static final String COMPONENT_URL = "component:/com.metaboss.sdlctools.services.codegeneration.BSCodeGenerator";
09
10     /** Generates code into the specified directory */
11     public void generate(String pDestinationDirectory) throws GenerationException;
12 }
13 }
```

Listing 2

```
01 import com.metaboss.sdlctools.services.codegeneration.BSCodeGenerator;
02 import com.metaboss.sdlctools.services.codegeneration.GenerationException;
03
04 .....
05 .....
06
07 try
08 {
09     // Get the context and lookup the component
10     javax.naming.Context lContext = new javax.naming.InitialContext();
11     BSCodeGenerator lGenerator = (BSCodeGenerator)lContext.lookup(BSCodeGenerator.COMPONENT_URL);
12
13     // Call the operation offered by the component
```

```
14     lGenerator.generate("c:/temp");
15 }
16 catch (javax.naming.NamingException e)
17 {
18     // Deal with the failure to find the component
19     handleException(e);
20 }
21 catch (GenerationException e)
22 {
23     // Deal with the component operation failure
24     handleException(e);
25 }
26
27 .....
28 .....
```

Listing 3

```
01 package com.metaboss.sdlctools.services.codegeneration.impl;
02
03 import com.metaboss.sdlctools.services.codegeneration.BSCodeGenerator;
04 import com.metaboss.sdlctools.services.codegeneration.GenerationException;
05
06 /** This is the default implementation of the component.*/
07 public class BSCodeGeneratorImpl implements BSCodeGenerator
08 {
09     /** Generates code into the specified directory */
10     public void generate(String pDestinationDirectory) throws GenerationException
11     {
12         // Do some useful code generation
13     }
14 }
```

PERSONALIZE YOUR WEB APPLICATIONS

Reusable components can eliminate personalization code from your applications

by Daniel Vlad

Personalization, a recurring requirement in most corporate Web applications, can be a very effective tool for streamlining Web applications and enhancing the Web user's experience. In many cases, personalization and security requirements go hand in hand; they can be dictated by corporate security principles and regulations that exist in banks, insurance companies, and any other organizations.

This article targets enterprise architects and developers who are willing to invest a little time to develop a set of easy-to-use, reusable personalization components that are powerful enough to meet the needs of many enterprise applications.

The personalization components that we describe here are based on rules declared and configured in an XML file. Personalizing Web applications with these components requires writing little or no Java code. Web pages are personalized using JSP custom tags, shifting the complex task of application personalization from the J2EE developers to Web page designers.

Two Typical Examples

1. Personalized Customer Account Maintenance Application

Suppose we're developing a Web application for viewing and maintaining customer account information. The account is defined by an account ID; customer information such as name, address, and phone number; and billing information such as a bank account.

Billing information is typically classified as sensitive information. In our example we require that only a subset of employees, account specialists and account managers, should be permitted to view the banking information. However, for operational reasons, all employees need to have access to nonsensitive account data, such as customer name and address.

In addition, we require that account maintenance functions, such as terminating or editing accounts, should be available to account managers only.

This example is typical for many applications that provide role-based Web access to corporate data.

2. Personalized Shopping Site

Here we want to develop a personalized shopping site

that recognizes high-value customers and rewards them for their loyalty. At checkout we want to display a coupon for 10% off the next order to customers who purchased \$1,000 worth of merchandise over the past 365 days (one year).

How would you develop these examples? Coding the personalization rules directly into the applications should not be an insurmountable problem for an experienced developer. However, in an agile enterprise rules change, sometimes very quickly. In the shopping site example we want to have the flexibility to change the promotion rules quickly, lowering the \$1,000 limit to \$500, for example. If this rule is hard-coded in your application, and especially if it's used several times in various parts of the application, you'll probably need to make substantial code changes, test the application for consistency, repackage it, and redeploy it on your application server. A framework based on components that centralizes the management of personalization rules and decouples these rules from the rest of the application suddenly sounds like a good idea.

Designing the Personalization Components

When it comes to personalization, different applications can have different requirements. In general, any user-related data can be used to personalize an application. It's important to recognize this early in our design so we can construct components that are as general as possible.

The central abstraction in our design is a personalization rule. We define a personalization rule as an object that can be evaluated as true or false; the outcome of the evaluation depends on user-related data and on the rule's configuration parameters. To allow for maximum flexibility and to avoid the hard-coding of the rule parameters in the application, the rules need to be configured externally.

To simplify JSP development, we want to be able to evaluate personalization rules on Web pages using tags from a JSP custom tag library. At runtime the appearance of the page will depend upon the outcome of the evaluations.



Daniel Vlad, PhD, is a senior consultant working for Highmark in Pittsburgh, Pennsylvania. Daniel leads the development of the corporate J2EE architectural framework, develops enterprise shared components, and provides guidance on the architecture, design and implementation of various applications.

daniel.vlad@highmark.com.

Personalization Rules

We represent a rule by an interface, PersonalizationRule. At a minimum the interface needs to provide a public method, boolean isRuleSatisfied(), which returns true if the user's data satisfy the rule, and false otherwise. To enable the rule to make a personalization decision, we need to supply it with user data, or at least with a mechanism for retrieving this data from an external database or directory. The HttpServletRequest provides us with the means for identifying users; we can either invoke its getRemoteUser() method (and rely on container-managed security) or we can retrieve the user authentication information stored in the request by third-party authorization systems. The user ID can then be used to look up or retrieve any user-related data from a database, LDAP, etc.

In addition to the isRuleSatisfied() method, the PersonalizationRule interface also contains setters/getters for the rule name and rule properties.

```
public interface PersonalizationRule {
    public String getRuleName();
    public void setRuleName(String ruleName);
    public String getProperty(String name);
    public void setProperty(String name, String value);
    public boolean isRuleSatisfied(HttpServletRequest req);
}
```

Implementations of this interface provide concrete behavior. Representing a rule by an interface gives us a great deal of flexibility in implementing personalization rules that match the requirements of the application.

XML Configuration File

An XML configuration file with the following structure is perfect for configuring personalization rules:

```
<personalization-rules>
  <rule name=" ..." classname=" ..." >
    <property>
      <name> ... </name>
      <value> ... </value>
    </property>
    ... other properties ...
  </rule>
  ... other rules ...
</personalization-rules>
```

For each personalization rule the XML document defines a rule name, and an implementation class name followed by a number of configuration properties in a name/value format.

Example: Personalization Based on Security Roles

As an example that will be used in the implementation of the Account Maintenance example, we can develop a RolePersonalizationRule class (see Listing 1) that is evaluated as true only if the user belongs to a supplied list of security roles.

To configure the rule we define a property named "roles"; the value of the property is a comma-separated list of security roles.

```
<rule name="rule1" classname="RolePersonalizationRule">
  <property>
    <name>roles</name>
    <value>role1,role2,role3</value>
  </property>
  ...
</rule>
```

To code the class we need to implement the methods specified in the PersonalizationRule interface. In the isRuleSatisfied method we parse the comma-separated list of roles using java.util.StringTokenizer, and check each individual role using the isUserInRole() method of the HttpServletRequest. If a match is found, we return true. If the user doesn't belong to any of the roles, we return false.

```
// Parse comma-separated list of roles
String roleToCheck= null;
StringTokenizer st=
    new StringTokenizer(rolesStr, ",");
while (st.hasMoreTokens()) {
    roleToCheck= st.nextToken().trim();
    // Check each individual role
    if (req.isUserInRole(roleToCheck))
        return true;
}
return false;
```

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

To complete the class we also need to implement the getters and setters for the rule name and properties. The properties can be stored in a Map instance.

Helpful design advice: You can place the implementation of the getter and setter methods into an abstract class and have all your rule classes inherit these methods, so you don't have to code them more than once.

Parsing the Personalization XML File

The parsing of the configuration file is performed by a utility, PersonalizationRuleParser (see Listing 2). The parse (String filename) method of this class is responsible for parsing the personalization XML file, and instantiating and configuring the rule objects.

Developers have a multitude of choices when it comes to parsing XML files. SAX and DOM are powerful and flexible APIs; however, they require a fair amount of coding. Programmers often utilize higher-level APIs to simplify their XML parsing code. To parse the personalization XML file we use the Jakarta Digester, a popular open source utility, very powerful in parsing XML and populating Java objects from XML documents.

Factory for Rules

We need to create a factory to manage the rules (see Listing 3). The factory invokes a method, initializeRules(), that invokes the PersonalizationRuleParser utility to parse the rules. The rules returned by the parser are cached in a static Map variable, rules. The initialization method is synchronized for thread-safety, thus guaranteeing that the parsing occurs only once.

```
private synchronized void initializeRules()
    throws PersonalizationException {
    if (rules == null) {
        parser= new PersonalizationRuleParser();
```

```
rules= parser.parse(fileName);
    }
}
```

The factory exposes a public method for retrieving a rule by name, getRule(String ruleName). If no rule object is found or if the object found does not implement the PersonalizationRule interface, the method throws an exception.

Personalization Custom JSP Tags

The classes created up to this point can already be used to make personalization decisions programmatically. However, to simplify the development of personalized JSP pages we need to create a number of custom tags.

Custom JSP Tags to Display User Attributes

A first class of tags that we can create contains tags that display user information, such as username, name, department, and address.

To display the username we need to develop a UserTag that extends javax.servlet.jsp.tagext.TagSupport and provides an implementation for the doStartTag() method (see Listing 4). The method will invoke the getRemoteUser() method of the HttpServletRequest object and it will write out the result using the JspWriter (see Listing 5).

The UserTag can be easily adapted to display other user attributes besides the username. We can use the username to look up the user data and print it out on the JspWriter using the same technique as in the code fragment in Listing 5. We can either create new tags for each user attribute or, even better, reuse the UserTag by enhancing it with an attribute to specify which user data to display.

Helpful design advice: To improve application performance and reduce the number of database or LDAP calls, you can retrieve the user-related data once and store it in a "User" object that can be placed in the user's Http session. Each rule has access to the request object and implicitly to the session, therefore it can retrieve the user object, get the user data, and use it in the evaluation of the rule.

Custom JSP Tags to Include or Exclude Web Content

The second category of tags that can be defined consists of tags for including (IncludeTag, see Listing 6) or excluding (ExcludeTag) Web content based on the outcome of the evaluation of the rules.

Both tags define a required attribute "rule" to specify the name of the rule to be evaluated.

The doStartTag() method of the IncludeTag contains code to include the body of the tag when the specified rule evaluates as true and to skip the body when the rule evaluates as false.

```
public int doStartTag() throws JspException {
    if (isRuleSatisfied()) {
        return (EVAL_BODY_INCLUDE);
    } else {
        return (SKIP_BODY);
    }
}
```

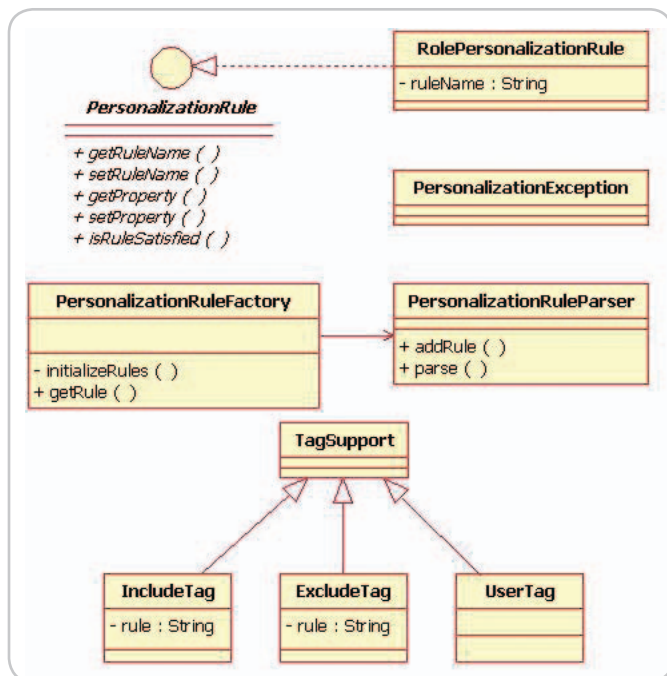


Figure 1 UML class diagram for the personalization components

The ExcludeTag is coded in a similar manner; the only difference is the reversing of the return values in the if/else statement.

Configuring and Using the Personalization Custom Tag Library

To use the custom tags we need to register them in a tag library descriptor (tld) file and declare the library in the JSP:

```
<%@ taglib uri="/WEB-INF/personalization.tld"
prefix="personalize" %>
```

Using the tags is straightforward. To print the username:

```
<personalize:user/>
```

To include JSP content (excluding is similar):

```
<personalize:include rule="MyRule">
... Web content to be included here ...
</personalize:include>
```

Using Personalization Rules Programmatically

Personalization rules can also be used programmatically, outside of the JSP pages. This comes in handy when we need to make personalization decisions in a servlet controller, for example, to redirect different groups of users to different Web pages. To evaluate a personalization rule programmatically, we just need to retrieve the rule from the factory and call the rule's isRuleSatisfied method.

UML Class Diagram

To summarize, Figure 1 shows a UML class diagram for the personalization components we have just created.

Implementing the Personalized Account Maintenance Example

It's time now to implement the examples described earlier in the article with our personalization components; we will start with the personalized account maintenance application. To implement the application, in the data access layer we retrieve the account information from a database and populate an Account transfer object (JavaBean). The bean is returned to the servlet controller (or Action, if you are using Struts), which binds the Account bean to the HttpRequest before forwarding the request to the JSP. The JSP retrieves the bean and displays its attributes on the page.

Personalizing the Application

The JSP needs to display the following Web content:

1. General account information that can be viewed by all authenticated users.
2. Bank account information that can be viewed only by account specialists and account managers. In the personalization XML file we define a personalization rule "ViewBankRule" of the RolePersonalizationRule type (we need to specify the fully qualified name of the implementation class):

```
<rule name="ViewBankRule"
```

```
classname="com.pers.RolePersonalizationRule">
<property>
  <name>roles</name>
  <value>Account Manager,Account Specialist</value>
</property>
</rule>
```

In the JSP we need to pass the rule to the personalize:include tag for evaluation:

```
<personalize:include rule="ViewBankRule">
  bank information here
</personalize:include>
```

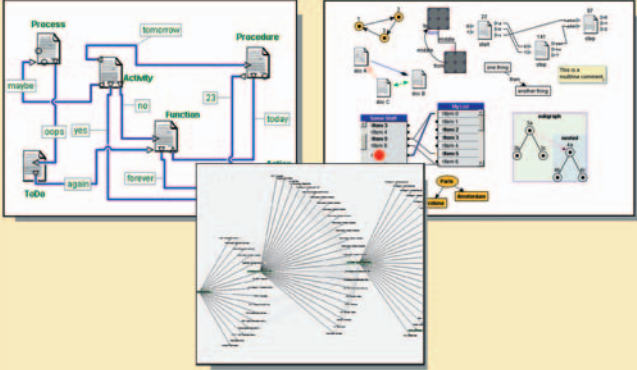
3. Account maintenance functions that should be displayed to account managers only. Similarly, we define a rule "EditAccountRule"; we configure the "roles" properties with the value "Account Manager"; and we surround the Web content with the personalize:include tag:

```
<personalize:include rule="EditAccountRule">
  account maintenance functions here
</personalize:include>
```

4. Finally, to display a personalized greeting in the JSP page, we use the user tag:

```
Welcome <personalize:user />
```


Build Incredible Interactive Diagrams with JGo™



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

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

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



Container-Managed Security Configuration

All personalization rules in this example are based on security roles. We need to configure the container-managed security environment for the Web application.

In the web.xml deployment descriptors we have to define the following security-related information:

1. Security roles needed by the application: employee, account specialist, and account manager.
2. Security constraints to exclude nonemployees from accessing the application and prohibit nonmanagers from accessing the URIs associated with the account maintenance actions.
3. Login configuration for the Web application, for example, basic authentication.

The last piece of the security configuration puzzle is to configure the Web container (Tomcat in our case) to use a “database” of usernames/passwords (security realm) for user authentication. To keep it simple, we use MemoryRealm, an in-memory representation of a database of usernames/passwords provided by Tomcat for development and test purposes. At runtime, Tomcat loads the usernames and passwords from the MemoryRealm XML configuration file (tomcat-users.xml) in memory, and uses this data to authenticate users.

Running the Application

After logging into the application and passing the security constraints enforced by the Web container, the user views a personalized application (see Figures 2–4).

Implementing the Personalized Shopping Site Example

To be able to selectively display the 10% discount coupon, we can define a rule named displayCoupon and implement it using a personalization rule class HighValueCustomerRule. The minimum purchase amount needed for the promotion to kick in and the time period in days are entered as configuration parameters in the XML file:

```
<personalization-rules>
  <rule name="displayCoupon"
        classname=" HighValueCustomerRule ">
    <property>
      <name>minOrders</name>
      <value>1000.00</value>
    </property>
    <property>
      <name>period</name>
      <value>365</value>
    </property>
  </rule>
</personalization-rules>
```

In the isRuleSatisfied method of the HighValueCustomerRule we get the userId (HttpServletRequest.getRemoteUser()) and use it to retrieve the total purchase amount for the specified period from the order database (the amount can also be retrieved by the controller in advance and cached in the user object). In the isRuleSatisfied we return true if the total is larger than the minOrder, and false if it isn't.

In the JSP we need to add the following:

```
<personalize:include rule="displayCoupon">

</personalize:include>
```

The rules are now easy to maintain. If we decide to lower the minimum purchase amount to \$500 instead of \$1,000, all we need to do is update the value in the personalization.xml file. Even radical changes that require replacing the entire HighValueCustomerRule with a different rule can be easily implemented when working within this framework.

If the same rules are used multiple times in an application, centralizing the configuration also ensures the consistency of the application logic.

Summary

Personalizing Web applications in a consistent manner can be challenging. Vendor personalization engines can be very powerful; however, they can also be extremely expensive and difficult to work with.

For most corporate Web applications, with the notable exception of portals, CRM, and other similarly complex applications, you might be better off building your own reusable personalization components or framework. The components shown here could provide a solid foundation for your personalization projects. Give them a try. ☺

References

- *Apache Jakarta Digester*: <http://jakarta.apache.org/commons/digester/>
- *Container-managed security*, see the *Servlet 2.3 specifications*: www.jcp.org/aboutjava/communityprocess/final/jsr053/
- *Tomcat*: <http://jakarta.apache.org/tomcat/index.html>
- *Tomcat 4.1 MemoryRealm configuration*: <http://jakarta.apache.org/tomcat/tomcat-4.1-doc/realm-howto.html#MemoryRealm>



Figure 2 Account Details page rendered to an Account Manager

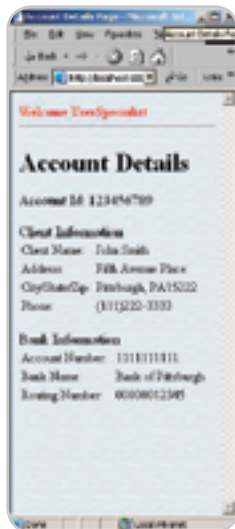


Figure 3 Account Details page rendered to an account specialist



Figure 4 Account Details page rendered to an employee

Listing 1:

```

public class RolePersonalizationRule
    implements PersonalizationRule {

    private String ruleName;
    private Map properties= new HashMap();

    public RolePersonalizationRule() {}

    public String getRuleName() {
        return ruleName;
    }

    public void setRuleName(String ruleName) {
        this.ruleName= ruleName;
    }

    public String getProperty(String name) {
        return (String) properties.get(name);
    }

    public void setProperty(String name, String value) {
        properties.put(name, value);
    }

    public boolean isRuleSatisfied(HttpServletRequest req){
        String rolesStr= getProperty("roles");
        if (rolesStr == null) return false;
        // Parse the comma-delimited list of roles
        String roleToCheck= null;
        StringTokenizer st=
            new StringTokenizer(rolesStr, ",");
        while (st.hasMoreTokens()) {
            roleToCheck= st.nextToken().trim();
            // Check each role.
            if (req.isUserInRole(roleToCheck))
                return true;
        }
        return false;
    }
}

```

Listing 2:

```

public class PersonalizationRuleParser {

    private Map rules= new HashMap();

    public void addRule(PersonalizationRule rule) {
        String ruleName= rule.getRuleName();
        rules.put(ruleName, rule);
    }

    public synchronized Map parse(String fileName)
        throws PersonalizationException {

        // instantiate Digester and disable XML validation
        Digester digester= new Digester();
        digester.setValidating(false);

        // Push this object to the Digester's object stack
        // making its methods available to processing rules.
        digester.push(this);
        // instantiate the PersonalizationRule object using
        // the classname attribute
        digester.addObjectCreate(
            "personalization-rules/rule",
            "PersonalizationRule", "classname");
        // set the rule name
        digester.addSetProperties(
            "personalization-rules/rule",
            "name", "ruleName");
        // Set the rule properties
        digester.addCallMethod(

```

```

        "personalization-rules/rule/property",
        "setProperty", 2);
        digester.addCallParam(
            "personalization-rules/rule/property/name", 0);
        digester.addCallParam(
            "personalization-rules/rule/property/value", 1);

        // call 'addRule' method when the next
        // 'personalization-rule' pattern is seen
        digester.addSetNext(
            "personalization-rules/rule", "addRule");

        // load file and start the parsing process
        try {
            ClassLoader classLoader=
                this.getClass().getClassLoader();
            URL url= classLoader.getResource(fileName);
            if (url != null) {
                InputStream in= url.openStream();
                digester.parse(in);
                in.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
            throw new PersonalizationException(
                "I/O errors while parsing the xml file");
        } catch (SAXException e) {
            e.printStackTrace();
            throw new PersonalizationException(
                "SAXException while parsing the xml file");
        }
        return rules;
    }
}

```

FREE* CD! (\$198.00 VALUE!)

Secrets of the Java Masters

Every *JDJ* Article on One CD!



— The Complete Works —

CD is edited by *JDJ* Editor-in-Chief Alan Williamson and organized into 33 chapters containing more than 1500 exclusive *JDJ* articles!
All in an easy-to-navigate HTML format! **BONUS: Full source code included!**

ORDER AT WWW.SYS-CON.COM/FREED

**PLUS \$9.95 SHIPPING AND PROCESSING (U.S. ONLY)*

©COPYRIGHT 2004 SYS-CON MEDIA. WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE. ALL BRAND AND PRODUCT NAMES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.



Only from the World's Leading i-Technology Publisher

Listing 3:

```
public class PersonalizationRuleFactory {

    public static final String fileName=
        "personalization.xml";
    private static Map rules;
    private PersonalizationRuleParser parser;

    public PersonalizationRuleFactory()
        throws PersonalizationException {
        initializeRules();
    }

    private synchronized void initializeRules()
        throws PersonalizationException {
        if (rules == null) {
            parser= new PersonalizationRuleParser();
            rules= parser.parse(fileName);
        }
    }

    public PersonalizationRule getRule(String ruleName)
        throws PersonalizationException {
        if (rules.containsKey(ruleName)) {
            Object o= rules.get(ruleName);
            if (o instanceof PersonalizationRule) {
                return (PersonalizationRule) o;
            } else {
                throw new PersonalizationException(ruleName
                    + " does not implement PersonalizationRule");
            }
        } else {
            throw new PersonalizationException(
                ruleName + " not found");
        }
    }
}
```

Listing 4:

```
public class UserTag extends TagSupport {

    public int doStartTag() throws JspException {
        HttpServletRequest request=
            (HttpServletRequest) pageContext.getRequest();
        String userid= request.getRemoteUser();

        // Print the userid to the page output writer
        JspWriter writer= pageContext.getOut();
        try {
            writer.print(userid);
        } catch (IOException e) {
            throw new JspException(e.toString());
        }
        // Continue processing this page
        return (EVAL_BODY_INCLUDE);
    }
}
```

Listing 5:

```
public int doStartTag() throws JspException {
    HttpServletRequest request=
        (HttpServletRequest) pageContext.getRequest();
```

```
String userid= request.getRemoteUser();
// Print the userid to the page output writer
JspWriter writer= pageContext.getOut();
try {
    writer.print(userid);
} catch (IOException e) {
    throw new JspException(e.toString());
}
// Continue processing this page
return (EVAL_BODY_INCLUDE);
}
```

Listing 6:

```
public class IncludeTag extends TagSupport {

    private String rule;

    public String getRule() {
        return rule;
    }

    public void setRule(String rule) {
        this.rule= rule;
    }

    public int doStartTag() throws JspException {
        if (isRuleSatisfied()) {
            return (EVAL_BODY_INCLUDE);
        } else {
            return (SKIP_BODY);
        }
    }

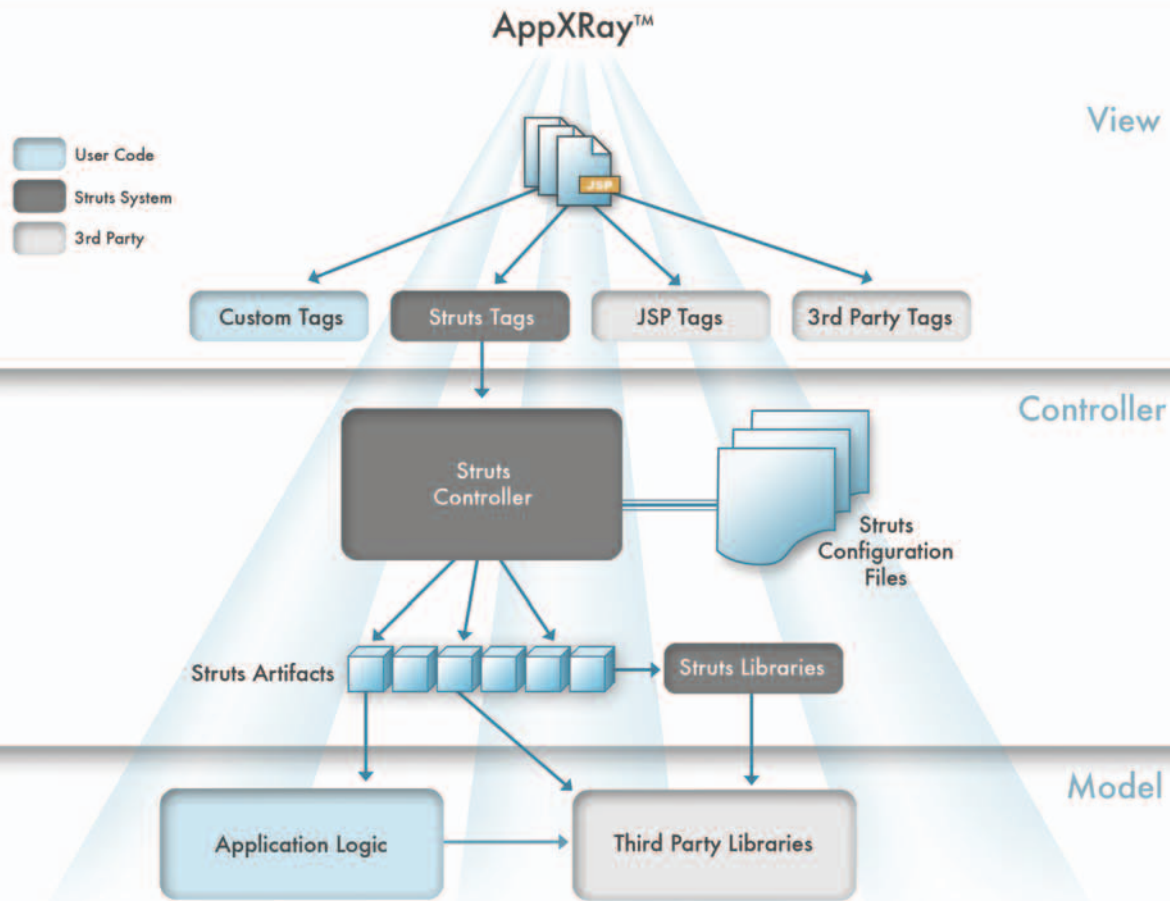
    public int doEndTag() throws JspException {
        return (EVAL_PAGE);
    }

    public void release() {
        super.release();
        rule= null;
    }

    private boolean isRuleSatisfied()
        throws JspException {

        PersonalizationRule ruleObj= null;
        if (rule == null) {
            throw new JspException(
                "Invalid use of the IncludeTag. rule=null");
        }
        try {
            PersonalizationRuleFactory factory=
                new PersonalizationRuleFactory();
            ruleObj= factory.getRule(rule);
        } catch (PersonalizationException pe) {
            throw new JspException(
                "Error while retrieving the rule " + rule, pe);
        }
        HttpServletRequest request=
            (HttpServletRequest) pageContext.getRequest();
        return ruleObj.isRuleSatisfied(request);
    }
}
```


Do you understand all the relationships in your web application?



Nitrox for JSP

Nitrox for Struts

Nitrox for JSF

NitroX™ AppXRay™ Does!

- JSP debugging including stepping in/out of JSP tags and unique JSP variables view
- Source and Visual JSP & Struts editors with knowledge of all web app layers
- Design time validation, consistency and error checking for JSP source, Tag Libs, Struts and Server-based Java code in a web app
- Eclipse & IBM WSAD, WSSD plug-in

Download a free, fully functional trial copy at: www.m7.com/d7.do



Copyright © 2004 M7 Corporation. All rights reserved. All M7 product names are trademarks or registered trademarks of M7 Corporation. Java and all Java based marks are trademarks or registered trademarks of Sun Microsystems. IBM, WSAD, WSSD are trademarks or registered trademarks of IBM.



Calvin Austin

Core and Internals Editor

Under the Hood of a J2EE Application Server

I recently had the opportunity to talk with many Java users about the current release and their general experiences with the platform. One of those developers told me that he didn't use J2SE but his J2EE VM sometimes caused problems.

Now most of you know that there is no such thing as a J2EE JVM. From my own experience with J2EE application servers, many do a good job hiding this from you. It can be difficult to work out which version of J2SE you are using, let alone know that J2SE is behind the scenes.

In this issue we are focusing on J2EE. If you are a core J2SE developer, there are some great tips and techniques that apply equally to J2SE and J2EE.

As a quick refresh, the J2SE platform contains the Java runtime and many of the core libraries that are used in your J2EE application. The Java runtime also contains client-side classes such as Swing. If those classes are not referenced by your application (which is normally the case), then they're not loaded at runtime. To compile Java programs use the javac compiler, which can be redistributed with the Java runtime and is also available in the Java developer kit.

One of the key technologies is the core XML library. In J2SE 5.0, the JAXP 1.3 library is bundled and includes support for XML 1.1, DOM Level 3, and SAX 2.0.2.

One new library in J2SE 5.0 that should be familiar to you is the addition of the Java Management Extension, JMX. JMX has been used in J2EE to monitor and manage J2EE applications using MBeans. The JMX

framework has been extended in J2SE 5.0 with a remote interface and also has some system MBeans and a mini MBeanServer that can be used to monitor low memory conditions. This same monitoring information is published via the SNMP protocol as well.

The other side of monitoring is application profiling. The core technology here has been JVMPI, the Java profiling interface. JVMPI is still supported in J2SE 5.0 but is superseded by the new JVM TI tools interface, which provides finer-grained and less intrusive profiling. Your existing profiling tools will require updates to support the JVM TI interface, but the benefits of improved profiling will repay the investment in newer tools.

The support for network-aware distributed objects is also provided in J2SE using technologies such as RMI, JNDI, and CORBA. There have been minor increments in J2SE 5.0 for each of these technologies. One change that makes J2EE developers' coding a little easier is that RMI no longer needs a separate stub creation step.

The Java language changes, of course, apply to J2SE and J2EE and the same javac compiler is used to build both types of applications. Later releases of J2EE will be able to take advantage of features like metadata to automate many tasks, including generating deployment descriptors.

For those developers who use JavaServer Pages, the newer javac compiler will work as before. There was a proposal to increase the maximum method length due to some side effects from some of the early JSP implementations. The 64K restriction

still exists but modern JSP compilers no longer run into that barrier.

Another area that is easy to overlook is database access. The JDBC framework is again delivered in J2SE. In J2SE 5.0 the addition of implementations for disconnected rowsets allows you to pass rows from your database and manipulate them without needing to maintain a live database connection. This technology even allows you to convert the database results to XML, provide updates using XML, and then resynchronize those changes at a later time.

It shouldn't be a surprise that the Java runtime is also responsible for the compilation of bytecodes and the threading framework. Many existing J2EE application servers provide support for a thread pool. In J2SE 5.0 the concurrency library also provides a choice of user space thread pools, which are portable across application servers.

J2EE application servers will also be able to provide more control over worker thread tasks by taking advantage of the new future tasks.

What does this mean for J2EE application developers? Well, regardless of the frameworks you use, J2SE 5.0 is going to bring two waves of improvements.

The first wave is updates to the libraries, such as XML, that are used by J2EE in the core platform. The second wave is to then exploit the advantages of the new features by the J2EE 5.0 platform and its components. I'm sure you'll agree that there is plenty to look forward to whether you consider yourself a J2EE developer, J2SE developer, or both. ☺

“It shouldn't be a surprise that the Java runtime is also responsible for the compilation of bytecodes and the threading framework”

A co-editor of *JDJ* since June 2004, Calvin Austin is the J2SE 5.0 Specification Lead at Sun Microsystems. He has been with Java Software since 1996 and is the Specification Lead for JSR-176, which defines the J2SE 5.0 ("Tiger") release contents.

calvin.austin@sys-con.com

Is your SOA really

agile?

Building, testing and maintaining SOA solutions requires a flexible and collaborative approach to diagnostics. Finger pointing and confusion occur when all parties do not have a complete and common understanding of a problem. Mindreef® SOAPscope® 4.0 connects developers, testers, support personnel, and operations by combining the testing and diagnostic tools for each discipline and making it easy to share complete problem data. SOAPscope is the only end-to-end diagnostic system for Web services.



Developers • Testers • Operations • Support

Mindreef SOAPscope 4.0

- Test** - Agile testing approach for both developers and testers allows what-if scenarios without coding or scripting. Includes the industry's most complete interoperability suite and the flexibility to really test the edge cases.
- Capture** - Easily capture and assemble complete problem data into SOAPscope's workspace. Capture SOAP messages off the wire, test cases, or reproduced problems. Capture a snapshot in time of a WSDL including all related resources.
- Share** - Exported workspaces support sharing among team members preserving complete problem data including messages, WSDLs, and notes. Sharing workspaces is a great way to support web service users.
- Solve** - Diagnostic suite of analysis, comparison and visibility tools simplifies the understanding of complex XML, yet are powerful enough for the industry's most protocol-savvy experts.

"SOAPscope has been invaluable in testing and diagnosing our enterprise scale Web Services. The new Workspace feature in SOAPscope 4.0 is an excellent way to support customers and collaborate as a team. A must have for every WS-Developer"

*Simon Fell
Senior Member of
Technical Staff at
salesforce.com and
author of PocketSOAP*

Try SOAPscope **FREE** at www.Mindreef.com/tryout



Developer Testing Is 'In'

An interview with Alberto Savoia and Kent Beck

Interview by Bill Dudney

A few weeks ago Agitar Software announced that Kent Beck had joined their team. I sat down and talked with Alberto Savoia, CTO, and Kent Beck, Agitar Fellow, to find out what prompted the move and what Agitar is up to that is so exciting.

JDJ: Kent and Alberto, why each other?

Beck: I think the primary motivation for the move is that Agitar is supporting very similar things that I have been working on for a number of years. You have a kind of leverage from working with a commercial company that you just don't have as an individual. The technology is also very interesting.

For 50 years it has been okay for the IT group to be sort of closed and not allow the business side to see what is going on until the end. The trend today is toward more transparency. Developer testing is one form of transparency or accountability. In the past business wrote a check and IT delivered something, but what was delivered was not always sufficient. Developer testing (and thus Agitator and Agitar) is one means of raising the level of transparency and accountability. The other thing is that developer testing makes the developer's job better; it lets you design better; it lets you do your job with confidence; and it lets you sleep better.

Savoia: Agitar probably would not exist if not for Kent and his contributions of XP and JUnit. Since XP is now cool, testing is now by inference cool. I always thought that developer testing is something we should have been doing all along and XP has made it legitimate for developers to test. More important, it is one thing to just say "you should be doing testing"; it's another thing entirely to

give developers a tool in the form of JUnit to help them do developer testing.

It's great to have Kent since he is the one who started this whole thing. Also, as we go forward in making developer testing ubiquitous, Kent's vision is going to be extremely influential in the way that we evolve Agitar.

JDJ: Is developer testing a fad?

Developer testing is something that should have been done from day one. For whatever reason developers abdicated this responsibility. Now with XP and JUnit, developer testing is making a comeback, but should have always been here.

Over the past year or so I've talked to literally hundreds of developers and development managers and none of them could make a good argument for not doing developer testing. The thing that comes up though is something I call the developer testing paradox, even though everyone thinks that developer testing is good. It's like motherhood and apple pie – everyone thinks it's a good idea but it's not nearly as widely adopted as it

should be. This is the paradox: Why is something so good practiced so infrequently? I believe it's because there are insufficient tools and processes to make developer testing more efficient and effective, and without these processes and tools it will be hard for developer testing to become as widespread as we believe it should be.

The other benefit we feel that you get out of developer testing is early and frequent feedback on your design. The first-order effect is great, a reduction in bugs, but the second-order effect of having better overall designs is probably even more valuable. With developer testing you get early feedback on the coupled (or not) nature of your design. If your design is highly coupled, you'll have complex setup and teardown. If your design is loosely coupled, you'll have a much easier time testing and a more flexible system.

JDJ: Is the developer testing that does exist sufficient?

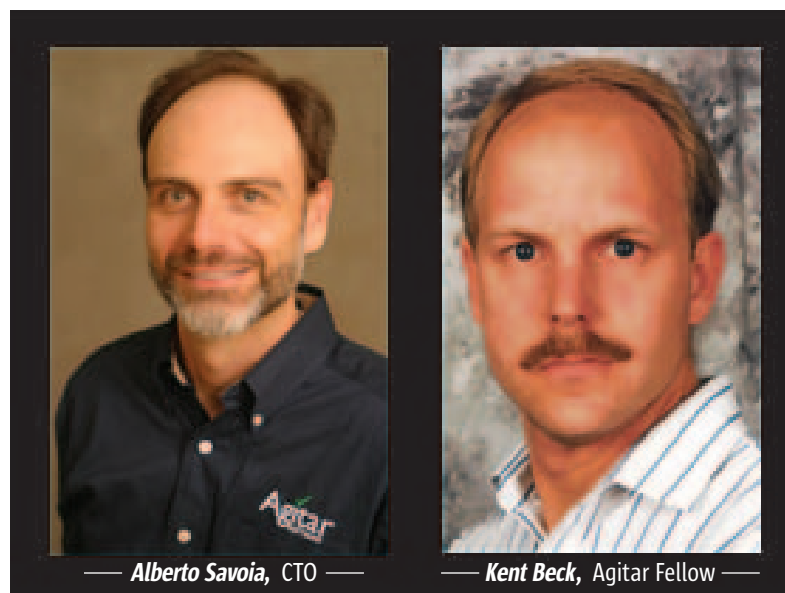
We have some empirical data that suggests that most IT shops that im-



Bill Dudney, JDJ's Eclipse editor, is a senior consultant with Object Systems Group.

He has been doing Java development since late 1996 after he downloaded his first copy of the JDK. Prior to OSG, Bill worked for InLine Software on the UML bridge that tied UML Models in Rational Rose and later XMI to the InLine suite of tools. Prior to getting hooked on Java, he built software on NeXTStep (precursor to Apple's OS X). Bill has roughly 15 years of distributed software development experience starting at NASA building software to manage the mass properties of the space shuttle. You can read his blog at <http://jroller.com/page/BillDudney>.

billdudney@sys-con.com



— Alberto Savoia, CTO —

— Kent Beck, Agitar Fellow —

Why Settle For "Sorta Close?"



Get The BPM That Fits

Reactor™ – the ideal Business Process Management (BPM) solution for Workflow Automation, Business Process Integration and Web Services Orchestration

Fits Within Your Architecture

J2EE-based, XML-driven, platform neutral

Fits Your Business Requirements

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

Fits How You Want To Buy

Flexibly priced, with source code access

Download your free evaluation copy at www.oakgrovesystems.com, or contact us at 1.818.440.1234. We can't wait to help you...

Declare Your Workflow Independence!™

oakgrove
systems

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

“Right now developer testing seems to be the exception rather than the rule. Our goal is to reverse that in the next three to four years”

plement developer testing see around 30% of their developers become “test infected.” Even if management decides to stop the unit testing effort, these developers would continue to do developer testing because they have realized the benefits.

The other 70% must have been immunized in their childhoods; the minute the pressure is off to build unit tests, they abandon developer testing and go back to writing lots of code with little testing.

The group who writes tests tends to write good tests, and their tests fail. Meaning that the good tests will find and prevent bugs. On the other hand, the group of developers who don't get test infected, write tests that almost always pass. These tests usually don't find bugs.

JDJ: Is this why we all too often see a system with a large set of developer tests fail when it's delivered to a testing group?

Exactly, this problem has a name: the pesticide paradox. Your code evolves to pass the unit tests; if you don't continue to increase the “dose” of the pesticide (the tests), pretty soon bugs adapt to the tests that you do have.

JDJ: Is developer testing increasing in the enterprise? What are you seeing among your clients?

The rarest of all is the company that decided to do developer testing, trained their whole team, and have been doing developer testing for quite a while, and are seeing success with this method. Usually these teams are led by people who are test infected; thus developer testing becomes part of the culture.

The largest group comprises companies that realize that testing is the correct thing to do. But when they try to implement they run into a set of problems. The primary one is that it takes a lot of time to test code and they don't know where that time will come from.

Finally, there are the companies that want to do developer testing but they just don't know where to start. Perhaps there are a few developers who are test infected but it's not something that is part of the team. I have yet to meet a team that says, “Hey, Alberto,

we have two weeks with nothing to do. We were thinking of trying out developer testing.” Developers always have something to work on, something to do. Moving to a developer-testing mindset is often difficult to do.

JDJ: In what ways does Agitator help solve these issues that you are seeing?

The main problem that I encountered when trying to instigate developer testing at Google was the time it takes to write the tests. If developers have to write 300–400 lines of test code to test 100 lines of source code, even though it's the right thing to do, they are making a large investment. The way that Agitator helps with this is by recognizing that much of testing is combinatorial in nature. For example, every if statement in your code needs two tests written. So writing tests by hand is great for particular test cases that you have thought about. Then you want to do exploratory testing and think about all the things that could happen.

I don't believe that code is the correct metaphor for testing. For instance, just like a spreadsheet is the correct metaphor for getting a bunch of calculations done or generating a graph. You don't care about all the stuff that goes on under the hood. You want to give the spreadsheet the input data, a list of formulas, and then have the result. A spreadsheet raises the level of abstraction to the things that you care about. Similarly Agitator raises the level of abstraction of the testing tasks to the components that are important. Those components are the test data and the assertions. The unnecessary distraction of the framework code is below the level of abstraction.

Along those same lines Agitator lowers the barrier to entry for developer testing thus making developer's lives more productive and more fun. In addition the tool also gives a means to measure what is going on inside the project. If you look at what everyone on the team is doing with Agitator and rolled that, you have a much more precise view of what is going on inside.

**JDJ: What level of metrics is the Agitator able to provide?**

We have spent a lot of time thinking about metrics. In fact our Dashboard product is all about metrics, and we have learned a lot of valuable lessons. When you want to institute developer testing, it's important to focus on positive metrics, in other words metrics that go up with better testing. Let me give you an example of one of the metrics reported by the Dashboard; we call it "Test Points." Whether you use JUnit or Agitator, every assert statement, for example, asserting that `add(2,2)` returns 4 is a test point. That is a metric that is positive. As that number grows, we can feel good. This metric is in contrast to traditional metrics like "the number of bugs found." If you are doing developer testing, presumably you won't find a lot of bugs with the tests because the tests help you prevent bugs in the first place.

Another metric that we use is percentage of classes or methods that have tests. Here the goal is very simple: have a test class for each class. There should be symmetry here. Measure how this metric grows as you achieve the goal that you have set for your group, then you can start to add a test for each method. As you achieve the goal you have for the number of methods touched by a test, you can get even more aggressive. However, this positive metric gives the team something positive to focus on and move toward. Instead of measuring your failures, you have favorable measurements to look at.

JDJ: Do you have any metrics regarding the use of Agitator over time? Something like, before developer testing there were so many bugs per 1,000 lines of code and after using developer testing the bug count fell to fewer bugs per 1,000 lines.

We have been doing developer testing on Agitator from the beginning and we currently have more than 20,000 test points for Agitator that run several times a day. It typically discovers a few bugs, and when we do major open-heart surgery on the code it finds many problems that we then fix.

Since we have been doing developer testing from the beginning, it's hard to offer a contrast. I can, however, point you and your readers to a recorded Webinar on our Web site (www.agitar.com) in which Jayson Minard of Abebooks.com talks about their use of Agitator. In a recent quarter they experienced zero downtime because of their use of developer testing and the Agitator. But even so, I still tend to go by gut feelings. The fact that I have 25,000 tests keeping the code clean and, if something goes wrong, I get red flags all over the place, makes me feel a lot better than a particular set of numbers.

“Since XP is now cool, testing is now by inference cool”

JDJ: Do you see the metrics generated by the Dashboard being misused, or is the test-point metric harder to misuse than coverage alone.

I believe in code coverage, but coverage without assertions is like testing a calculator by pushing the buttons for three hours and never looking at the display. I can say that the calculator did not catch on fire but I can't say if it functioned correctly. For example, when an application starts up, you could see as much as 30% code coverage, but you have not discovered that it stated correctly since just starting does not run any assertions. What we do is create a mapping between code coverage and assertion coverage. The theory is that a method is "covered," but if there are no assertions for that method, the coverage does not get counted in the test points metric. Sure it is possible that the method has been tested indirectly, but without a direct test I can't relax and sleep well at night until I have that level of testing. Furthermore if your code cannot be tested at the class and method level your code might not be unit testable and therefore has some likely coupling problems.

JDJ: How does the Agitator fit into a typical developer process?

We ran an experiment (documented at www.developertesting.com) where we did a project test first using JUnit. The interesting thing is that when you do this kind of testing, the developer tends to focus on the positive tests, i.e., $2 * 2$ is in fact evaluating to 4, which is natural. However, we always fired up Agitator to help us think about the corner cases that we didn't consider during our typical testing. We found two interesting results. We expected the first; namely that the corner cases that Agitator found were indeed busted, e.g., if I pass a string that is too long/short to this method, the code breaks. The other outcome was that Agitator forced us to think through further refactorings that we did not previously consider. This process is documented in the series of articles on this project at developertesting.com (Agitator-driven refactoring). Using JUnit we thought about "localized" refactorings on particular classes, but in bringing in agitation we saw areas where we should do wider refactorings. It showed us more of the global-level dependencies that were not readily apparent without the agitation process.

JDJ: Where are you headed with the tool set? What's next?

Our vision and one of the reasons we hired Kent Beck is that right now developer testing seems to be the exception rather than the rule. Our goal is to reverse that in the next three to four years. Our product goal is to make Agitator more and more integrated with the development process to assist in the movement toward a major shift into developer testing in the industry.

We are also developing new and interesting ways to display the mountain of information that we gather with the Dashboard. For example, you can see "risky" classes. Risk is a combination of the complexity of the class and the dependencies on that class. With the Dashboard we are headed to more summarization of the information we gather as well as giving people some data on a successful project so that they have something to compare against. ☺

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.



Joe Winchester
Desktop Java Editor



Sticks and Stones

While at lunch with colleagues recently I overheard four very able Java developers swapping horror stories of the kit they'd cut their teeth on as junior programmers. One had used a Sinclair ZX-81 with 1K of RAM and a black and white TV and a tape recorder in lieu of a hard drive. Things were so bad with the memory that the screen buffer was used to store program data. That story was trumped by tales of a Commodore 64 where after it was discovered that the built-in hard drive had its own processor, it was used to offload program work to create true symmetric multiprocessing on a box that was never designed for this.

Both stories came from wizened old developers convinced they'd won the mantle of "hardship programming in my youth." Ironically the college kid in the group played the five-ace hand with tales of taking a games console and stripping it down to a Unix box and then running it as his home server. Apart from feeling I was observing a 21st century version of Monty Python's "Four Yorkshiremen" sketch (www.phespirit.info/montypython/four_yorkshiremen.htm), I felt slightly worried by the allegories being drawn and implied by the dialog.

Without necessarily intending it, two messages were being conveyed. One was that you had to have struggled with basic tools and underpowered hardware to get a job done as a necessary rite of passage on the path to becoming a true programmer, and the second was the syllogism that if you continue to use sticks and stones to program you are somehow a superior programmer than someone who doesn't.

I encounter quite a lot of the latter attitude in my day job where I develop tools whose purpose is to help people write Java GUIs. When talking to users, some of them enjoy the obvious benefits of being able to preview their

screen as they write it and handle components by selecting and manipulating them in WYSIWYG touchy-feely viewers. Others will rightly point out that the complexity of their GUI is beyond what the builder can cope with and that's a good reason not use it. Others seem almost threatened by what is basically a fairly entrenched idea of a tool creating code for you and they'll come up with every excuse possible why they shouldn't use it. Arguments vary from the code not conforming to their particular style, which, apart from often degenerating into an academic argument about how fields should be named or methods structured, also shows a certain intransigence against being able to adapt to a new style. It exhibits a belligerent and generally frowned upon trait in programmers to be inflexible when dealing with code authored by others without feeling the compunction to rewrite it.

The favored vestige of the non-IDE folks is often to drop down to a text editor such as vi or Emacs and gain some kind of Luddite satisfaction from editing raw files and firing off build scripts. I used to admire people who enjoyed coding with such tools similar to the way you might enjoy listening to classical music played on period instruments. After a while, however, it just starts to sound flat and hollow and in denial of all that has been learned since. The problem is being able to recognize the genuine aficionado of someone who is using his favorite antique editing experience because it genuinely makes him productive and able to focus on his job without helpful wizards and code assist, versus the one who is trying to copy the master and thereby fake credibility that masquerades his inability to focus on his job.

At a recent presentation I was giving on an IDE feature that makes it easier to write code, I had to deal with a

heckler who told me how superior vi was and how we couldn't deal with his particular key bindings. The irony was that he wasn't from the wizened penguin bumper-sticker brigade who are the stereotypes of such attitudes – he was a fresh-faced, recent college grad and had brought his bellicose attitude almost as a badge of honor to parade.

Let's take cars, for example – my attitude is that when they break I take them to the mechanic; what's fun isn't so much tinkering with the engine myself and gaining some kind of machismo pride in doing so, it's the journeys I take in it and what I do when I arrive. Likewise with writing software – the purpose is to create a good user experience for someone else who wants to solve a particular problem in a more efficient way. I once had to explain to a customer why we were late shipping a particular software release and he replied that we were just polishing the inside of a tin can and he didn't care. He was right – we were upgrading operating system releases and migrating to a new language version mid-release cycle. However, there was no business value to it and we'd just taken our eye off the target and onto our navels.

Is the problem with software and tooling one of a master craftsman with his favorite chisel and simply that people are reluctant to change something that makes them most productive, or is it just that people have a built-in desire to belong to a herd and gain social acceptance from their tribal peers from where they can collectively mock progress and other languages and technology changes as being for the folks on the other team? Is the super-league of programmers occupied by folks who take apart game boxes and have wireless networks in their kitchen, or is it by those who would rather play some fun games on the box it was designed for and then enjoy a nice meal in the kitchen afterward? ☺

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

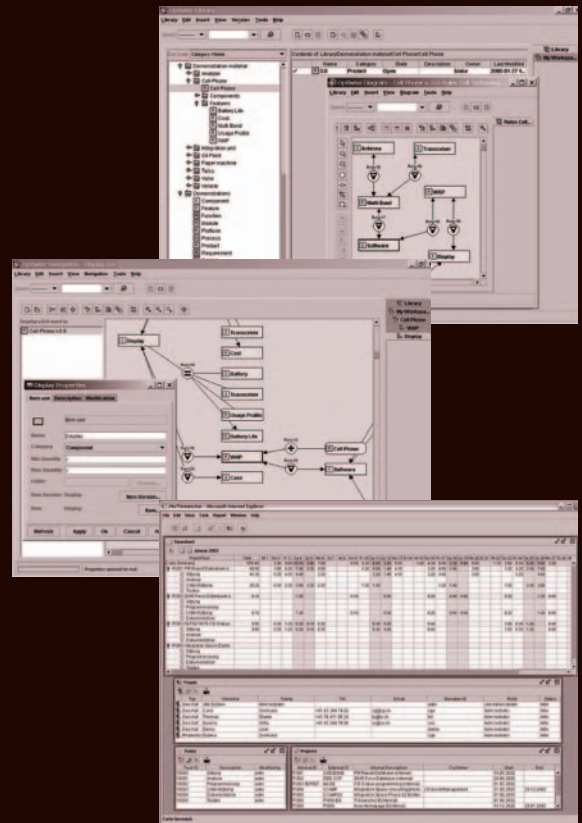
joewinchester@sys-con.com

ULC

Rich Thin Clients for J2EE

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

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



canoo.com

Canoo Engineering AG

<http://www.canoo.com/ulc/>

Download your free trial today!

Providing a Complete Data Services Layer

Exclusive JDJ Interview with **Dennis Leung**,
Vice President of Oracle Application Server, TopLink Development



Interview by Jeremy Geelan

For those involved in the maintenance and programming of databases, object-relational (O/R) mapping and TopLink have been almost synonymous for 10 years. An innovator in the ORM space for an entire decade, TopLink was started in 1994 as an independent company and was acquired by Oracle from the defunct WebGain in June of 2002. As Oracle customers have expressed their needs in this area, TopLink has become integrated not just into Oracle's app server but also with JDeveloper, the Application Development Framework (ADF), and the BPEL database adapter. In this interview Dennis Leung, vice president of development for Oracle TopLink, talks with JDJ about persistence, O/R mapping, EJB 3.0, Web services, SOA, and more.

Since Oracle acquired TopLink, its profile has diminished and it seems to have been sucked into a black hole. What happened? Is there a future for TopLink at Oracle?

I certainly don't think it has been sucked into a black hole! What has happened is that we've gone from being the primary product of a small company to a very large company with a broad, rich portfolio of products. It's been a successful transition into Oracle, and the global resources at our disposal have allowed us to reach more customers than ever. We are still very much committed to the TopLink product. The future is bright for TopLink as we bring the results of our recent efforts to market. Oracle, with its leadership in relational database technology and its commitment to J2EE, is an ideal home for TopLink.

It's also been good for our customers. Even with our 10-year track record we faced the scrutiny all small companies face. Enterprise customers need to know you have long-term viability, especially if they are going to use you as a key component in their application. Being part of Oracle removes that concern.

As you say, TopLink is celebrating its 10-year anniversary and that's unusual for a software product. You've been involved with TopLink since its inception, during which time the software development landscape has changed tremendously – to what do you attribute the longevity of TopLink?

I can think of a few reasons, but the primary one is that we've listened to our customers and have addressed their

requirements that are coming from real projects. A great ORM product is not something a group of people can sit in a room and plan out. You have to live through the demands of real projects hitting difficult challenges. TopLink is a 10-year reflection of having thousands of customers' experience building enterprise applications for a wide cross section of industries.

In our rapidly evolving and highly competitive industry, to survive this long a company needs to provide value to customers and continue to provide value over the years. We've been able to foresee and adapt to changes in technology. We started in Smalltalk, moved to Java, gone from client/server to server-side J2EE/EJB, and are already receiving good feedback on our Web services and SOA efforts.

Finally, the core team has remained together. Most of the TopLink team has been here for five to seven years. We've been fortunate to have added some really talented developers over the past few years as well. This continuity balanced with the influx of new ideas has served us well.

How about the longevity of Dennis Leung? ;-) Is Oracle still a stimulating company to work for?

As with any new situation, there's a certain degree of apprehension and uncertainty. This was certainly true for me personally as I had not worked for a large company like Oracle for about eight years. I can unequivocally say after two and a half years, I really enjoy working here. I've learned a great deal as the mandate for my group is much broader with involvement in areas such as BPEL, Web services, Oracle JDeveloper, ADF, SOA, and JAXB.

Since day one, my management chain has been very supportive of our group and our responsibilities have increased. For example, the Oracle representatives on EJB 3.0 and JAXB 2.0 are from the TopLink team and we've put a considerable amount of work into those JSRs. Oracle is a serious player in the Java universe and that provides tremendous opportunities and interesting challenges.

What is the biggest misconception about persistence products?

Over the years the developer community has become more familiar with the problems persistence products address; however, there still are a number of misconceptions.



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

The difficulty of the fundamental O/R problem that is being addressed is often underestimated and developers don't think they need a persistence solution. On the surface it seems easy: hand code some SQL, do a bit of data conversion, build some objects – no problem. However, it quickly grows into a very complex problem as you try to address issues such as inheritance, caching, querying, concurrency, advanced database features, and clustering. This is when the developer realizes that a persistence solution is needed.

Also, because the only visible part of a persistence product is the GUI tool, many people think it's just a design-time tool, when in fact it has a considerable presence at runtime.

Other big misconceptions are that persistence products can't handle difficult problems – the SQL generated is inefficient and hand-coded JDBC will run faster. Over the years we have proven time and again that this is not true.

From your many years of experience, what are the most challenging problems your customers experience when building applications using O/R mapping technology?

The first challenge that customers face is with their development processes and corporate culture. A lot of projects don't facilitate enough communication between software development and the database management. In some cases, these responsibilities are spread across different sites. Without good communication and mutual respect for how Java applications are developed and how databases are managed, the technical impact of impedance mismatch is compounded.

Technically speaking, the most challenging problems usually revolve around performance. In a J2EE application, normally the most expensive operation that can occur is a call across the network to the database. You want to minimize these calls and do them intelligently. This is where using a proven solution is key because a persistence layer needs to provide lots of flexibility on how to bring back data depending on the volume and volatility of your data.

Another challenging area is dealing with concurrency, particularly in clustered applications. The goal is to provide as close to "turn-key" clustering as possible. It should be trivial to scale an application from two to four to many more nodes. There is a lot to consider though – locking, caching, and refreshing are just a few of the issues.

Does TopLink support multiple databases? Is it as performant as you would ideally like?

Yes, TopLink has always supported multiple databases. This is also true for J2EE application servers. TopLink runs on any J2EE-compliant application server. Since we were acquired by Oracle, one of the misconceptions about TopLink is that we would only work on the Oracle Java stack. This is far from the truth. One of the fundamental benefits of TopLink is flexibility, which includes running on any app server and using any major relational database vendor. In fact, our daily regression test suite runs TopLink

on Oracle OC4J, WebLogic, and WebSphere against Oracle, DB2, Sybase, and SQL Server.

Performance is something that we continually try to improve. We have a lot of performance-related features in TopLink but you can never allow yourself to become complacent in the area of performance.

Given that you are still around, do you consider the product a success?

I measure success based on whether organizations choose TopLink and successfully build and deploy their applications. Based on these criteria, TopLink is very successful.

Looking at the competition, open source is growing in popularity and mindshare; what impact has it had on TopLink?

We've always treated our competition with respect and open source is no different. There have always been a fair number of competitors in this space and this has provided plenty of inspiration for us to continue to innovate and evolve TopLink.

The result of having many vendors and open source products in this space is that the persistence market itself has grown. I think more people are better educated about this problem domain.

We used to have to go into accounts and convince developers and management that O/R is a difficult problem and they could actually save time and money by leveraging a third-party product rather than building the infrastructure themselves. We don't need to do this as often anymore.

A slightly related impact that open source has had is that we've put more emphasis on clarifying the availability of our products to developers. Like all Oracle development products, free, full functionality, nonexpiring versions of TopLink are available for download from our site. This isn't a new policy, but one we've been more explicit about.

How do you differentiate your offering from those of your competitors? What are the best metrics for measuring the relative effectiveness of different persistence solutions?

TopLink's differentiating factors revolve around the richness of our feature set as a result of being in this space for 10 years. TopLink is the only product to support POJOs and CMP entity beans, allowing them to intermix as well. This means our customers who are currently using either POJOs or entity beans are well positioned to transition to EJB 3.0. We have tremendous flexibility in providing options for areas such as performance tuning, caching, querying, concurrency, clustering, database features, and platform support. Developer productivity is enhanced with powerful GUI-based tools and the ability to choose the standalone TopLink Workbench or fully integrated mapping functionality in Oracle JDeveloper.

On a broader scale we've grown from being a point ORM solution to adding capabilities to support XML and EIS



Technically speaking, the most challenging problems usually revolve around performance”



DESKTOP



CORE



ENTERPRISE



HOME

data sources. We believe these are among the key features to providing a complete data services layer that is required to address the diverse needs of today's enterprise applications.

The best metrics are based on the actual requirements of a particular application. Identify a difficult, deep vertical portion of your application and see if the persistence solution can address those needs. Can it effectively map the object model? How easy is it to define queries? Can you leverage the database features the DBA has utilized? Can it scale and perform? TopLink has experienced the most success where customers have had in-depth and challenging evaluation criteria.

I keep coming back to providing value for customers and there are a number of dimensions in providing value. I've talked about the obvious product ones, but being part of a large established company like Oracle has allowed us to offer value in ways we couldn't before.

The number one factor is the peace of mind that accompanies company viability. Another aspect of this is that we have the ability to extend support for older versions of TopLink that in the past we would have long discontinued. We can service our customers much better with 24x7 technical support, global consulting, and local account teams; our runtime exceptions are translated into 22 languages – all the benefits you would expect from a software company the size of Oracle.

There has been a lot of controversy and debate in regards to standards in the O/R space with JDO and EJB 3.0. What is Oracle's position on these standards?

We are fully behind the EJB 3.0 specification and our expert group representative, Michael Keith, has played a pri-

our O/R solution to XML data sources. It's JAXB compliant, but goes beyond the spec to provide developers with complete control of how an object model is mapped to an XML Schema. Unlike basic JAXB, you aren't forced to use the object model that is generated.

We've also used TopLink to build a database adapter that allows BPEL engines to easily define and utilize database functions as part of a business process.

Both of these are examples of how TopLink is evolving beyond a point ORM solution to provide a broader set of persistence capabilities for diverse applications accessing a variety of data sources.

How does Microsoft's offering in the ORM space compare to what J2EE offers.

The difference is night and day. Even though J2EE and .NET face the same challenges of O/R mapping, Microsoft has no solutions available.

Back in 2001, Microsoft previewed something called ObjectSpaces, which is essentially their ORM solution. It still hasn't been released and I believe it has even been deferred from the delayed Longhorn release, so who knows when it will actually be in production. Even though ObjectSpaces is taking a long time to come to market, its very presence has limited the third-party market as Microsoft has indicated it will eventually be available.

This is a concrete example of how .NET is not ready for "Enterprise-level prime time." Yes, I'll be the first to admit that not all apps need an ORM solution, but a great number of enterprise apps do and .NET offers very little for those apps.



XML is a versatile data exchange format, but it's not the ideal structure from a programmatic perspective"

mary leadership role on that committee. The recent move of having a group of JDO experts join the EJB committee is a positive step and clearly provides industry support for O/R mapping in Java to be standardized with the EJB 3.0 specification.

The feedback from the early drafts of the EJB 3.0 specification has been great. The POJO-based lightweight persistence model is one that we've endorsed for a long time and, with our support for both EJB 2.1 and POJO, TopLink customers are well positioned to transition to EJB 3.0.

Oracle, BEA, and IBM are all strongly committed to Web services and SOA. Where do you see TopLink's technology fitting in this picture?

We've leveraged core TopLink technology to address the unique requirements such as XML binding that materialize with Web services and SOA applications. XML is a versatile data exchange format, but it's not the ideal structure from a programmatic perspective.

TopLink's object-XML capabilities bring similar levels of flexibility and control developers have come to expect from

What does 2005 hold in store, in your view? Will ORM be even more critical to building manageable applications? Will the rise of "API-agnostic" O/R mapping continue?

For the foreseeable future, object-oriented technology and relational databases will be fundamental parts of IT solutions, so I think ORM solutions will continue to play a critical role. EJB 3.0 will solidify in 2005 and, with the improvements it brings, I think the industry will converge on that standard.

As Java and J2EE become more commonplace in IT, the scope of a persistence product must grow beyond ORM to address all the needs of a true enterprise, which include the integration with disparate data sources. This is where I see the future of persistence solutions moving. In addition to O/R mapping, they need to also provide services such as O-XML mapping, data access for BPEL, data binding for Web services, distributed cache management, integration with legacy systems, and event data management. All in all, it's an exciting time to be in the persistence space. ☺

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.



The World's Leading i-Technology Publisher

Unbreakable Java

by Thomas Smits

A Java server that never goes down

Developers using Java on clients or in small projects may not believe that there is a fundamental problem with Java's robustness. People working with huge applications and application servers written in Java know about the problem but may doubt that it's possible to build something like an unbreakable Java architecture. Some may even remember the White Star Line promising that their ocean liner *Titanic* was unsinkable; an iceberg in the North Atlantic proved them wrong and demonstrated that there is no such thing as an unsinkable ship. Is it really possible to build a Java application server that never goes down?

It's All About Isolation

The key to understanding robust Java is isolation, isolation, and isolation. Robust applications, especially robust application servers, require a high level of isolation between users. It's not acceptable that an error occurring while processing one user's request may affect all users connected to the system. The complexity of software systems makes it impossible to develop software that is completely free of errors, so errors will always happen. Only isolation can provide real robustness by limiting the impact of errors.

The design of the Java Virtual Machine ignores the painful lessons operating system vendors have learned in the past 40 years. The concepts of processes, virtual memory management, and different protection modes for kernel and user code can be found in all modern operating systems. They focus on the question of isolation and therefore robustness: an application with errors cannot affect the other applications running in the system.

In contrast, Java follows the all-in-one-VM paradigm: everything is processed inside one virtual machine running in one operating system process. Inside the VM, parallelism is implemented using threads with no separation regarding memory or other resources. In this respect Java has not changed since its invention in the early nineties. The fact that Java was originally invented as a programming language

for embedded devices may explain this approach.

There Is No Isolation in Java

Java does not have a problem with isolation; there is virtually no isolation at all. Java tries to avoid dangerous concepts like manual memory management (this is like taking some of the icebergs out of the ocean) and it can't be denied that it provides at least some isolation concepts, but a Java Virtual Machine is still easy to break. For example, class loaders make it possible to partition an application into parts that cannot see and access each other directly, which provides some isolation. Going back to our nautical example from the very beginning, this is exactly what was supposed to make the *Titanic* unsinkable: the ship consisted of separate compartments and water pouring into the ship was supposed to be stopped by the bulkheads separating the compartments – unfortunately the iceberg was too big and way too many compartments filled

up with water. In terms less familiar to the sailor but more familiar to the developer: all the fancy isolation built with class loaders does not help if you have memory leaks, threads running amok, or even bugs in the VM.

SAP's Approach to Isolation

SAP's ABAP application server – the powerhouse underlying enterprise-scale R/3 business solutions – was based on the concept of process isolation from the very beginning. It consists of a dispatcher and a bunch of work processes handling the requests. The work processes are normal operating system (OS) processes and the OS provides a high level of isolation for free. The dispatcher guarantees that in one moment exactly one user request is processed by each work process. In case of a crash, only the user currently processed in the crashing process is affected. All other users continue their work and the operating system takes care of the resource cleanup.



Thomas Smits has a degree in business administration and economics with a focus on business information technology. His first contact with Java was 8 years ago. Since then, he's been eating, sleeping, and drinking Java. He did development projects for German Rail (Die Bahn), Brenntag, and other companies. Thomas authored a course on Java Web technology for Sun Microsystems and has done a lot of customer-specific training on Java. Since 2002 he is a development architect in the SAP NetWeaver team.

thomas.smits@sap.com

Why Is It Called VM Container?

The technology behind the Always On Java initiative is called VM Container and the name suggests that there is something like a virtual machine and a container housing it. Right!

The name is based on the fact that the ABAP application server already contains a lot of interesting and battle-tested services that can be reused to build a robust Java server. The components were re-shaped and now provide the container that hosts the Java Virtual Machine. The VM was licensed by SAP and modified to seamlessly integrate into the container and to provide additional features like sharing technologies and enhanced supportability.

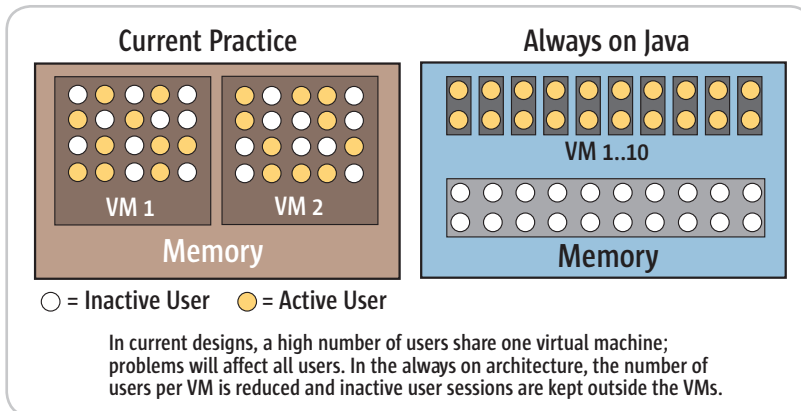


Figure 1 Virtual machines

To overstress the ocean liner example a little: the ship is not split up into compartments but every passenger gets its own ship (a separated process) with some guide (the dispatcher) taking care that all sail the same course and don't hit each other. Using this architecture, an iceberg (a severe error) may still hit one of the ships but it will affect only one passenger.

One passenger per ship sounds weird. Giving each passenger his or her own private dining room and engines seems to be a huge waste of resources. Two things can be done to handle the resource issue. First, it is possible to let the passengers share the ship with some others without meeting them at any time. Some invisible mechanism moves the sleeping passenger out of the ship, storing him or her somewhere outside and puts another active passenger into it, taking care that only one active passenger is in each ship at any moment. The second way to address the resource problem is to share as many resources as possible between the little ocean liners.

In the ABAP application server, the state of the user – often called user context – is not stored inside the process but in a shared memory area accessible to all work processes. This allows attaching the user context to a free work process when the next request arrives. Attaching user contexts is a very fast operation because no data is copied.

The ABAP virtual machine (yes, ABAP is executed on a virtual machine) was designed from the very beginning to store user contexts in shared memory. All infrastructure (the engines, the dining room) is written in C and able to deal with user contexts being moved between the work processes too.

User Isolation in Java

SAP's VM Container technology transfers the ABAP isolation concepts to the Java arena. The first step is to increase the number of virtual machines and therefore reduce the number of users handled by each VM. Having a hundred instead of a thousand users assigned to a VM makes a difference in case of a crash, but still affects too many users. Decreasing the number of affected users further without increasing the number of virtual machines requires some extra magic.

Normally less than 10 percent of the users connected to a system are actively sending requests; the others are thinking about their next action or typing in some data at the front end (thinking users). Keeping the user state (user session in Java terms) in a memory area outside the virtual machine allows reestablishing the sessions of all thinking users in case of a crash. This reduces the

number of affected users in our example to only 10 or one percent of the thousand users (see Figure 1).

The technology used to keep the sessions outside the virtual machine is called Shared Closures (see sidebar for details). The session state of a user is saved to shared memory after his or her request was processed. This guarantees that the shared memory contains a backup of the session state of at least all thinking users and that the data is accessible to all virtual machines. In case of a crash, another virtual machine can copy the user state from shared memory to its local memory and continue processing the user's requests without the user even noticing.

Memory Diet for the VM

The drawback of the described approach is that you have more virtual machines, each of them eating up some memory. This requires extra measures to keep the memory footprint of the VMs low; they must be put on a diet. This problem is addressed by Shared Classes.

The memory consumed by Java classes can become quite large in real-world applications. Shared Classes is a technology built into the Java Virtual Machine that shares the runtime representation of the classes, including the native code generated by the JIT compiler, across all virtual machines on one physical box. The classes exist only once in memory, reducing the overall memory consumption of the VMs.

In addition to the session backup explained earlier, Shared Closures can be utilized to reduce the memory footprint of a virtual machine. Configuration data and other application or server-wide information can be shared between VMs. Mapping the data from shared memory will provide access to it without consuming memory in each VM.

Don't Forget Supportability

Providing a high level of robustness through isolation is half the battle, but robustness without supportability is not sufficient. If something goes wrong in the application server, support personnel must be able to track down and resolve the problem easily.

The virtual machine used in the VM Container has been improved regarding supportability. One of the most interesting features is the ability to switch dynamically into debugging mode and vice versa. The switch can be initiated from the inside (using Java code) or from the outside (using administrative tools). Normally, Java application servers need dedicated debugging nodes because the Java virtual machine must be switched

Shared Closures

One of the key features of the VM Container technology is the Shared Closures API. It provides a semantic similar to serialization but with a new and very fast implementation. This technology enables middleware developers to share Java objects between virtual machines running on the same computer. For the application developer, high-level APIs based on Shared Closures are available, for example, providing caching or configuration management.

The name Shared Closures already implies that not only single objects but the whole transitive closure of objects reachable from one root object is shared. This behavior is like Java serialization except the operations are faster and a special mode of operation, called mapping, is supported.

A Shared Closure is created or updated by providing a reference to the root of an object tree to the API. The content of the tree is copied to the shared memory while the objects inside the virtual machine remain unchanged.

An exiting Shared Closure can be used in two different ways:

- **Copy:** The objects in the Shared Closure are copied to the heap of another VM. The objects become normal local objects and can be modified (see Figure 2).
- **Map:** The objects in the Shared Closure are not copied but only mapped into the address space of the virtual machine (see Figure 3). This operation is very fast in comparison to copy, because no data is transferred. Especially no extra memory is consumed for the mapped objects. The objects mapped into the address space are read-only.

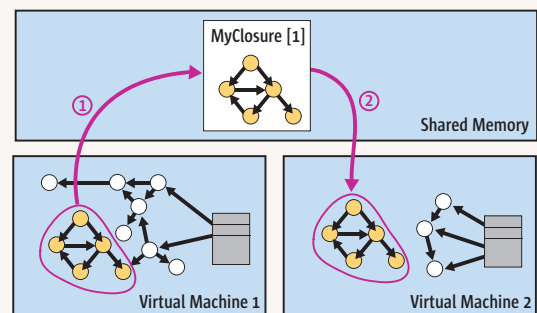


Figure 2: Create and copy of a Shared Closure

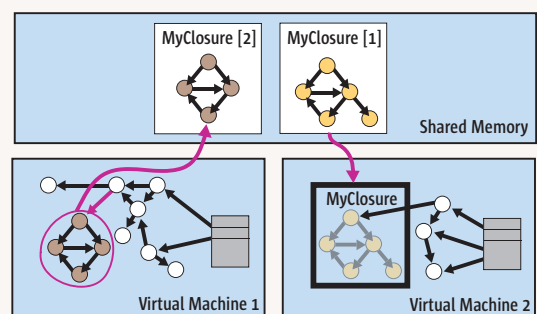


Figure 3: Versioning and mapping of a Shared Closure

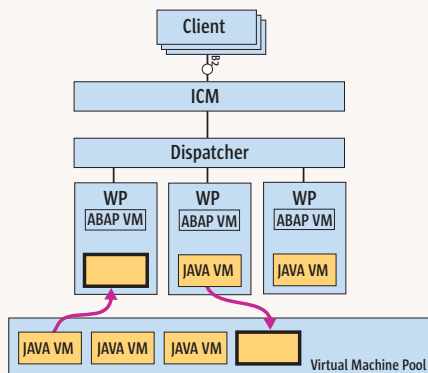
An implicit versioning mechanism takes care of the fact that some VMs may have mapped a version of a Shared Closure when another VM wants to publish an update. All previously mapped closures remain unchanged, whereas new map requests provide the new version. A distributed garbage collector removes all old versions that are no longer used.

Mapping objects from Shared Closures is the best mode of operation for caches and configuration data that rarely changes. Copying the data of a Shared Closure is used to implement session failover or messaging mechanisms.

A Peek into the Labs: Full User Isolation

In the development labs at SAP, work is in progress on a solution that goes beyond the approach described in this article: it merges the Java and the ABAP world. Both virtual machines run together in one work process and full user isolation is provided for ABAP and Java programs: in one Java Virtual Machine, only one user request is processed at a time.

A new paradigm was implemented called Process Attachable Virtual Machines. It decouples the VM from the process and makes it a lightweight memory image that can be moved between processes. Using VM templates, new virtual machines for the pool can be created with nearly no runtime effort. VM templates are available that contain a fully bootstrapped virtual machine, including the application server and the deployed applications. Using VM templates offers a way to create new virtual machines for the pool instantaneously.

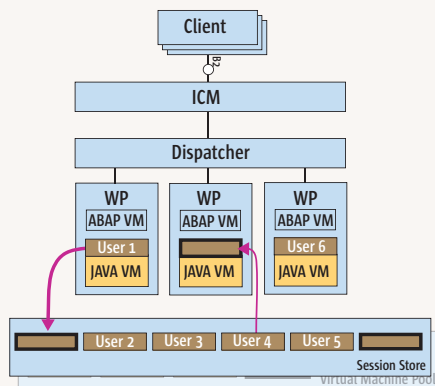


The virtual machine pool contains process attachable VMs that can be assigned to work processes on demand.

The number of work processes can be configured in a way that guarantees that the working set of all processes fits into the machine's main memory (although the memory is usually too small to hold all VMs at the same time). The number of virtual machines in the pool is normally higher, to take into account situations where a virtual machine does blocking I/O or other operations that don't use the CPU. In those cases, the VM is temporarily detached to free the process for new requests.

The operating system schedules preemptively between the processes but the virtual machines are moved in and out of the processes on a semantic base (semantic scheduling). This dramatically reduces the problem of thrashing because the working set is only changed after a user request is finished. Controlling the semantic scheduling is easy because the VMs are not operating system processes but attached to processes and detached on demand.

The session state of the users is kept in a special shared memory area accessed via the Shared Closures technology. The VM and the user session are separated after each request. Therefore the VMs can be used independently of the user sessions; there are no sessions sticky to a special VM except in the moment when a request is processed.



User sessions are kept in shared memory and copied to the virtual machine before the request is processed and copied back after the request is finished.

Virtual Machine

The virtual machine used for the VM Container is based on a Sun CDC/Hotspot VM. It was originally designed for embedded devices, making it very lightweight and easy to port to new platforms. Having a VM with a low memory footprint is important because the isolation approach of the VM Container will increase the number of parallel running VMs. You may imagine the VM Container as a cluster of Palm Pilots if you like.

into debugging mode at start-up. Using the VM Container, debugging is possible at any time, even in productive systems. A sophisticated rights management restricts which parts of an application or server a developer can debug. This prevents misuse of debugging capabilities in production environments.

Besides debugging, the monitoring capabilities of the VM can be used to obtain granular statistics about the running server. The monitoring is built in a way that does not affect the performance of the running application until explicitly switched on.

Summary

The VM Container technology offers improved robustness through isolation. The isolation is provided by reducing the number of users handled in parallel in one virtual machine. Saving the user's session state in a shared memory area improves the failover characteristics of the application server. Advanced sharing technology helps to reduce the memory footprint of the virtual machines. Improved monitoring and debugging support makes it easy to detect and fix problems at runtime.

References

- Tanenbaum, A. (2001). *Modern Operating Systems (2nd Edition)*. Prentice Hall.
- Byous, J. "Java Technology: the Early Years": <http://java.sun.com/features/1998/05/birthday.html>
- SAP Web Application Server Components: <http://help.sap.com>
- Kuck, N., et.al "SAP VM Container: Using Process Attachable Virtual Machines." Java Virtual Machine Research and Technology Symposium, San Francisco, August 2002.
- J2ME CDC HotSpot Implementation Overview: <http://java.sun.com/products/cdc-hi/overview.html>

VIEWPOINT

– continued from page 6

Indeed, Spring and Hibernate – the leading so-called “alternative” frameworks – are challenging the J2EE programming model while embracing the J2EE technology platform. This is a critical distinction. With Spring particularly, you get the power and maturity of the J2EE stack with a simplicity comparable to that of the .NET programming model. And you get it all with less cost and considerably more business leverage (choice).

How is this possible? When you step back and look at J2EE, there is a lot to it. J2EE consists of:

1. A standard set of enterprise services addressing typical infrastructure needs, including transaction management (JTA), dynamic user content (servlets/JSP/JSPF), database access (JDBC), service lookup (JNDI), asynchronous messaging (JMS), management (JMX), and remoting (RMI/Web services).
2. A standard programming model for tapping into the power of the above services, gluing the individual pieces together into a consistent software delivery platform.

We love the enterprise services. Nobody can match the power and maturity of the J2EE technology stack. However, today’s pragmatic developer isn’t so keen on the EJB programming model that builds on these services. It’s complex, invasive, dated, and overengineered for most developer needs. It’s not consistent. I have to jump through hoops just to run my business logic in different environments, for example. I incur the costs of JTA when all my application needs is a single database. The list of unnecessary costs goes on.

Enter the “alternative” frameworks. Spring, in particular, has given our community a framework that not only makes it easier to tap into the power of J2EE, but captures best practices on what services to use when given your business requirements. The result? More developers, architects, and managers are getting smarter about the infrastructure they need for the given job at hand. Developer productivity is up.

For over a year I’ve personally leveraged the Spring Framework as the base architecture for my development projects. I now treat J2EE infrastructure as a separate concern, one fully decoupled from the business logic (“core meat”) of my application. Spring gives me the power to choose which deployment environment and technologies are most appropriate given the complexity of the domain problem at hand. That puts me in command – if all I need is a Web container to power a Web app with a single data source, a solution like Tomcat is the best cost. For middleware-intensive applications that require messaging, global transactions, and remoting, a higher-end application server is worth the investment. In all cases, my programming model stays simple and consistent, grounded in my customer’s problem domain.

I can’t say it enough, J2EE is better than ever – for the consumer. I read success story after success story from developers working on projects with products like Spring and Hibernate. They’re leveraging them in all kinds of environments and application servers to support demands on all scales. Today is a great time to be developing enterprise applications in Java. It’s a great time to be a consumer. We’ve got the technology, the platform, and the community – it’s only going to get better. Who can stop us now? ☺

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	4
Borland	www.go.borland.com/j6	831-431-1000	9
Business Objects	www.businessobjects.com/dev/p7	888-333-6007	27
Canoo Engineering AG	www.canoo.com/ulc		49
Common Controls	www.common-controls.com	+49 (0) 6151/13 6 31-0	35
Dice	www.dice.com	877-386-3323	29
ESRI	www.esri.com/develop		7
EV1 Servers	www.ev1servers.net	800-504-SURF	15
Google	www.google.com/cacm	650-623-4000	31
ILOG	jviews-info-kit.ilog.com	800-for-ILOG	17
Information Storage & Security Journal	www.issjournal.com	888-303-5282	61
InstallShield	www.installshield.com/jdj	800-809-5659	19
IT Solutions Guide	www.sys-con.com/itsolutions	888-303-5282	53
M7	www.m7.com/d7.do	866-770-9770	41
Microsoft	msdn.microsoft.com/visual		Cover II
Mindreef	www.mindreef.com	+1 603 465-2204	43
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	37
Oak Grove Systems	www.oakgrovesystems.com	818-440-1234	45
Parasoft Corporation	www.parasoft.com/achievequality	888-305-0041	23
Quest Software, Inc.	http://www.quest.com/jdj	800-663-4723	Cover IV
SAP	www.sdn.sap.com		11
Software FX	www.softwarefx.com	800-392-4278	Cover III
WebRenderer	www.webrenderer.com		47
Web Services Edge 2005 East	www.sys-con.com/edge	201-802-3069	58-59

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.



Web Services Edge 2005 East

International Web Services Conference & Expo

- New for 2005:*
- Colocating with LinuxWorld Conference & Expo
badges give access to both shows
 - Guaranteed Minimum Attendance
3000 Delegates
 - New, Hot Sessions & Seminars

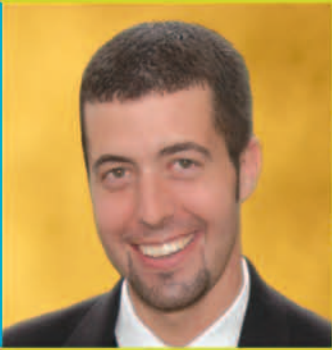
The Largest i-Technology Event of the Year!



Tuesday, February 15
11 a.m.

Matt Ackley
Senior Director, eBay
Developers Program
Topic:
Web Services for
eCommerce

Keynote Speakers & Featured Guests



Wednesday, February 16
11 a.m.
Ari Bixborn
Director, Web Services
Strategies, Microsoft Corporation
Topic:
Indigo and the future
of Web Services



Anne Thomas Manes
Burton Group
**Application Server
Shoot-Out Facilitator**

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

For Exhibit and Sponsorship Information:
Jim Hanchrow, 201-802-3066, jimh@sys-con.com

Register Now! Hot Sessions & Seminars

- Ensuring Web Services Interoperability
- Web Services Standards: Going Behind the Mask
- The Role of Policy in Web Services Integration – It's More Than Just Security
- Four Abilities SOA will lack without a Registry
- Driving SOA Governance
- BPEL Best Practices from Real-World Projects
- SOA: From Pattern to Production
- Lessons From the Front Line – Building Interoperable Web Services
- Developing E-Commerce Applications with Web Services
- Developing Enterprise Class Web Services
- Orchestrating FORCEnet Engagement Packs with BPEL for Web Services
- CPI: A Global Integrated Problem Tracking and Resolution System using Java Web Services
- Using Service-Oriented Architecture and Web Services to Issue Business Licenses in the District of Columbia
- Developing Web Services with Eclipse
- The Interoperability Challenge of Web Services Security Standards
- Securing Web Services with WS-Security
- Anatomy of a Web Services Attack
- Using a Mobile Phone as an SSO Authentication Device in SOA Solutions
- XML Content Attacks
- XACML and Agnostic Authorities

Register Today: www.sys-con.com/edge2005east/registermew.cfm



Come and see what everyone is talking about...

The Application Server Shootout!

Find out which server best fits your performance and ROI metrics

Join us in delivering the latest, freshest, and most proven Web services solutions...

At the *Fifth Annual Web Services Edge 2005 East – International Conference & Expo* as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics.

Exhibit in *The Web Services Pavilion* and see thousands of buyers!

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

3-Day Conference & Education Program features:

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

PRODUCED BY
SYS-CON
EVENTS

www.sys-con.com/edge

Sponsored by:



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

From Within the Java Community Process Program



Onno Kluyt

The votes are in

Welcome to the December edition of the JCP column! Each month you can read about the Java Community Process: newly submitted JSRs, new draft specs, Java APIs that were finalized, and other news from the JCP. This time around I'll cover the recent elections for the Executive Committees and four new J2ME-related JSRs.

The Envelope Please!

During October and November the annual elections for the two Executive Committees take place. Each year the elections are managed by PriceWaterhouseCoopers to ensure an impartial execution of the proceedings.

In October the JCP membership ratified the nominations that Sun had put forward, resulting in the appointments of Apache, Borland, and Nortel to the SE/EE EC, and NTT DoCoMo, RIM, and Samsung to the ME EC. The main criteria that Sun uses in selecting JCP members for nomination are concentrated on maintaining (and sometimes improving) broad market representation and geographic diversity – a little more on that in a moment.

The second phase of the elections is the open election on JCP members who nominated themselves for the open seats. This year for the ME EC two places were available, and the SE/EE EC had three available seats. Out of 10 candidates the JCP members voted to elect Google, JBoss, and Intel to the SE/EE EC. From the seven candidates for the ME EC, Intel and Orange France won the election. Congratulations to the winners, and for the others hopefully better luck next time.

Together with the outcome of the ratification vote, there are some

interesting new dynamics emerging. On the SE/EE EC there is better representation for the JAIN and OSS/J efforts in the JCP through the appointment of Nortel. I'm glad that Apache accepted our invite to continue their participation at this level in the JCP, so that open source viewpoints are present during our meetings. And with the election of the JBoss Group there will be additional perspective brought in from this section of the community. Then there is the relative newcomer to the Java community in the form of Google, albeit represented by a familiar face in the person of a spec lead of old, Josh Bloch (welcome back!). On the ME EC there will be more balance between manufacturers' interests and providers' interests through the appointments of NTT DoCoMo and Orange France. I'm also pleased to see an increased Asian presence on this EC through Samsung as well as NTT DoCoMo.

The newly elected EC members took office on Tuesday, November 30.

New Activities in the J2ME Technology Space

The J2ME technology section of the JCP is busy and hard at work. The four new JSRs that I'll discuss in this month's column are all J2ME focused.

JSR 256, Mobile Sensor API, was submitted by Nokia and approved by the ME EC. The JSR proposed to build an optional package to communicate with and manage sensors from the mobile device. There are many different sensor types (heart rate monitors, thermometers, cameras, microphones, etc.) that may connect to mobile devices using a variety of protocols. This JSR plans to provide a common approach to interacting with sensors for CLDC 1.1-based devices.

Nokia also submitted the proposal for the Contactless Communication API or JSR 257. Contactless communication, both one-way and bidirectional, may be based on RFID (Radio Frequency Identification), NFC (Near Field Communication), or bar codes. These mechanisms are used to exchange relatively small amounts of data between devices, or transport that data to a device from product packaging, for example.

The third new JSR, Mobile User Interface Customization API or JSR 258, was also approved by the ME EC but not until after the submittal of various voting comments by the EC members. The Spec Lead proposed to define a generic approach for the customization of a device's user interface, whether the interface is based on LCDUI (MID-P), AGUI (JSR 209) or AWT from Personal Profile (just examples). One approach it will use to achieve this is through the creation of a common vocabulary where the vocabulary maps to the customization properties of the device's user interface.

JSR 259, Ad Hoc Networking API, was proposed by Siemens. This JSR plans to specify an API that enables communication between mobile devices in a peer-to-peer networking environment, and enabling these devices to dynamically join ad hoc networks. The JSR will provide facilities such as service discovery and registration, and capability inquiry. The API would give access to various dynamic communications technologies, such as ZeroConf, UPnP, and JXTA.

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

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

onno@jcp.org

Subscribe Today!

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

FREE RESOURCE CD INCLUDED!

information
STORAGE + SECURITY
journal
www.ISSJournal.com

In this issue:

- IP Storage
- SAN NAS
- Disaster Recovery
- Enterprise Security
- Infosecurity

PREMIER 2004 VOL.1 ISSUE 1

Microsoft
RSA Security
SecureID for Windows

SAN NAS

Emerging technology trends and market maneuvers



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 i-Technology Publisher

Why Web Applications Can be Problematic and Unreliable



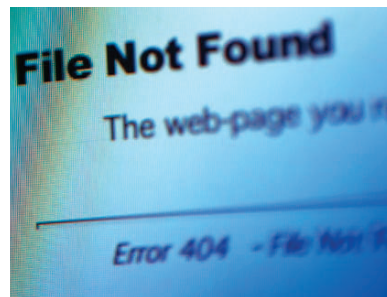
by Coach K. Wei

It's no surprise that the common perception is that Web applications are unreliable and problematic. Users often experience "404," "resource unavailable," and "network unavailable" errors or even a mysterious application error telling them to "retry the application later." The truth is, a fundamental source of all these problems is the HTTP communication layer of the Web.

The Internet was initially designed for presenting and sharing hyper-linked documents in the form of Web pages. Therefore, the communication layer is based on the HTTP "Request/Response" model, which adequately serves the purpose of page browsing. However, the Internet has since evolved far beyond simply supporting browsing activity and is now being utilized as an interactive platform for supporting mission-critical enterprise applications. However, in this context, there are still problematic Web browsing assumptions made – especially with regard to the messaging layer – when it comes to Web application development.

First off, the Web's messaging layer does not support guaranteed message delivery. When a user submits a request to the server, whether this request will actually arrive at the server or not is unpredictable. If there is a network problem (either with the ISP or within the corporate network), there is a good chance the request will be lost. However, this is not always a problem for Internet browsing, as the user can always click the link a second and third time if the first URL request is lost. Although this seems like a basic example, it's a serious problem for mission-critical enterprise applications. The point is, it's not out of the realm of possibility that a multi-million dollar transaction can literally be "riding on the line."

Another factor to consider is that the Web's messaging layer does not guarantee the order of message delivery. If the user submits two requests in a row, there is no guarantee that the first request will arrive at the server before the second request. Again, while this is not necessarily a problem for browsing Web pages, the result of a later request can be dependent on an earlier request when using the Internet for business applications. A random ordering of message delivery makes the application's behavior unpredictable – a pattern that many Web application users are familiar with.



Further, the Web's messaging layer does not support server-initiated or server-push communications; it supports client-pull only. In a client-pull only model, the server works like a phone that never rings. Obviously this is not a problem for browsing because the server simply responds to page requests. However, many enterprise applications require the server to initiate interactions. For example, a stock trading application needs to push the latest stock price from the server to the end user. To side step this problem, developers typically use client polling, but this significantly increases the server/network load and therefore decreases application performance.

As discussed, when considering developing and deploying Web applications, the messaging layer's unreli-

ability and limited functionality are fundamental problems that must be seriously considered. A potential solution to the message reliability issue is to implement message queuing on both the client and server side. Thus, if a message failed to be delivered due to a temporary network problem, the message can be retrieved from the queue and re-sent once the network is available. Message queues also help guarantee the order of delivery. To get around server-push limitations, some developers leverage client-polling techniques. Although it is possible to develop a solution in-house to address some of these issues, the technical challenge can be significant and the cost to test and maintain the solution can be high.

An alternative approach is to investigate some commercially available application platforms that have built-in solutions to all these problems. Some Rich Internet Application (RIA) platform solutions now available have a built-in capability for reliable messaging and real-time server push. Such solutions not only deliver a richer application experience, but also dramatically improve the application's robustness and reliability by providing a stronger communication layer – a critical factor when considering the Internet as a means of housing enterprise-level applications.

As the Web continues to become more widely adopted as a mission-critical enterprise application platform, it's even more imperative that developers truly understand and take into consideration its flawed communication model. There are now RIA solutions available to help developers overcome these limitations. Otherwise, traditional Web applications will not only frustrate end users and IT staff, they will also introduce significant problems that disrupt everyday operations. ☺

ENTERPRISE CHARTING SOLUTIONS

Need More Than A Pretty Face?



For Serious Enterprise-Level Data Visualization & Analysis

Multi-Platform
Ease of Use
Outstanding Support
...and A Pretty Face!

Extensibility
Performance
Scalability

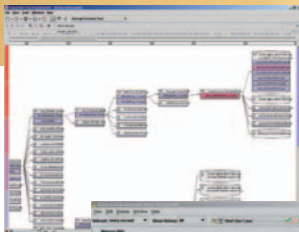


Chart FX

www.softwarefx.com



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



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



JProbe – a performance tuning toolkit for Java developers.

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

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



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