

JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

January 2002 Volume: 7 Issue: 1

JAVADEVELOPERSJOURNAL.COM

Download
HP-AS 8.0
FREE



www.hpmiddleware.com/download

From the Editor
by Alan Williamson pg. 5

Letters to the Editor
pg. 7

J2EE Application Security
by Timothy Fisher pg. 28

Core J2EE Patterns
by Dan Malks pg. 32

Industry Commentary
by Nigel Thomas pg. 74

"Ross Report"
pg. 100

Cubist Threads
by Blair Wyman pg. 114

RETAILERS PLEASE DISPLAY
UNTIL MARCH 31, 2002

\$5.99US \$6.99CAN



SYS-CON
MEDIA

written by
Boris Lublinsky

The Key









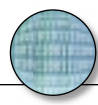





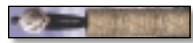

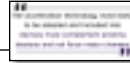


to Superior EJB Design

Decrease network traffic in EJB implementations

page 20

Will .NET Jolt Java ?

Rick Ross visits Microsoft
pg. 100 ▶▶▶▶▶▶▶▶

	J2EE Testing: What's It to You		Jonathan Maron
	<i>An overview of the CTS for J2EE component developers</i>		12
	Journeyman J2EE: Getting Focus()ed – and a Quick JavaScript Lesson		Charles Arehart
	<i>A win-win situation</i>		38
	Regex: Using Regular Expressions in J2SE 1.4		David Weller
	<i>Even a regex neophyte should have no trouble with the pattern</i>		46
	Feature: Using Assertions in Java		Jonathan Amsterdam
	<i>An easy way to build confidence in your code</i>		52
	Multithreaded Access: Synchronizing Java Threads		Vishal Goenka
	<i>A simple and elegant solution</i>		60
	Dynamic Tools: Evolutive Java Applications		Alvaro Schwarzberg
	<i>Enable functionality-driven software architecture</i>		66
	Feature: J2ME Benchmarking		Carl Barratt & Glenn Coates
	<i>Evaluating the performance of a JVM objectively</i>		78
	Embedded Computing: Hardware Accelerators for J2ME Come of Age		Ron Stein
	<i>Java gets a needed boost</i>		84
	Q&A: Ask Doctor Java		James McGovern
	<i>Prescriptions for your Java ailments</i>		90

sonic

www.sonic.com

zerog

www.zerog.com

bea
www.bea.com

INTERNATIONAL ADVISORY BOARD

- CALVIN ALSTIN (Lead Software Engineer, J2SE Linux Project, Sun Microsystems)
- JAMES DUNCAN DAVIDSON (JavaServlet API/JMP API, Sun Microsystems)
- JASON HUNTER (Senior Technologist, CollabNet)
- JON S. STEVENS (Apache Software Foundation)
- RICK ROSS (President, JavaLobby)
- BILL ROTH (Group Product Manager, Sun Microsystems)
- BILL WILLETT (CEO, Programmer's Paradise)
- BLAIR WYMAN (Chief Software Architect IBM Rochester)

EDITORIAL

- EDITOR-IN-CHIEF: ALAN WILLIAMSON
- EDITORIAL DIRECTOR: JEREMY GEELAN
- J2EE EDITOR: AJIT SAGAR
- J2ME EDITOR: JASON BRIGGS
- J2SE EDITOR: KEITH BROWN
- PRODUCT REVIEW EDITOR: JIM MILBERY
- FOUNDING EDITOR: SEAN RHODY

PRODUCTION

- VICE PRESIDENT, PRODUCTION AND DESIGN: JIM MORGAN
- ASSOCIATE ART DIRECTOR: LOUIS F. CUFFARI
- EXECUTIVE EDITOR: M'LOU PINKHAM
- MANAGING EDITOR: CHERYL VAN SISE
- EDITOR: NANCY VALENTINE
- ASSOCIATE EDITORS: JAMIE MATUSOV
GAIL SCHULTZ
- ONLINE EDITOR: LIN GOETZ
- TECHNICAL EDITOR: BAHADIR KARUV, PH.D.

WRITERS IN THIS ISSUE

- JONATHAN AMSTERDAM, CHARLES AREHART, BILL BALOGLU, CARL BARRATT, JASON BRIGGS, KEITH BROWN, GLENN COATES, TIMOTHY FISHER, VISHAL GOENKA, BORIS LUBLINSKY, DAN MALKS, JOHN MARON, JAMES MCGOVERN, AJIT SAGAR, ALVARO SCHWARZBERG, RON STEIN, NIGEL THOMAS, DAVID WELLER, ALAN WILLIAMSON, BLAIR WYMAN

SUBSCRIPTIONS:

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT

SUBSCRIPTION HOTLINE: SUBSCRIBE@SYS-CON.COM

COVER PRICE: \$5.99/ISSUE

DOMESTIC: \$49.99/YR. (12 ISSUES)

CANADA/MEXICO: \$79.99/YR. OVERSEAS: \$99.99/YR.

(U.S. BANKS OR MONEY ORDERS). BACK ISSUES: \$10/EA., INTERNATIONAL \$15/EA.

EDITORIAL OFFICES:

SYS-CON MEDIA 135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9600

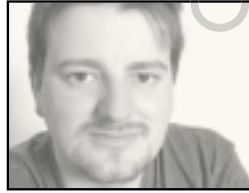
JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is published monthly (12 times a year) for \$49.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. POSTMASTER: Send address changes to: JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

© COPYRIGHT:

Copyright © 2002 by SYS-CON Publications, Inc. All rights reserved.

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

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



ALAN WILLIAMSON EDITOR-IN-CHIEF

Scandalous Propaganda:

'Twenty-Eight Times Faster than J2EE!'

It's the start of a new year; what fruits will our computing orchard serve up this season? This time last year the industry was excitedly preparing us for how Web services would take over. Sun was gearing up for their Sun ONE announcement in February after Microsoft had begun filtering out information on what their .NET was really about. With 12 months now in the time bank, I can truly say I have not seen any major change. Just a lot more people than usual, particularly vendor companies, talking a great game.

A couple of months ago I introduced the notion that the whole Web services revolution was a marketing gimmick, dreamed up to put a new spin on old products. I purposely went a little over the top just to test the water. I was interested in what you thought about the whole Web services movement and I'm happy to report that you were not backward in coming forward. I received a lot of e-mail and I thoroughly enjoyed the dialogue with you over this. I can safely report that approximately half of the responses agreed with what I had to say; 35% thought I was missing the point and 15% believed I was so far off base that I didn't have a clue what I was talking about. Coincidentally, the majority of that 15% were sales and marketing people, which I have to admit made me chortle somewhat.

My major difficulty with Web services, I believe, stems from the name. I just hate the name. It says nothing and in my opinion undervalues the whole movement. Web services is not a revolution but more of an evolution – taking technologies and attempting to coordinate them into one coherent solution as opposed to having formats and standards fight for their market share. Take e-mail, for example. Turn the clock back 20 years, or even as little as 10, and you had e-mail systems that

could not communicate with one another without a specialized gateway that would translate the e-mail. With the standardization of SMTP as the underlying mail transport protocol, we can now e-mail the majority of the world without too much worry. So, while I applaud the movement, I've spent the last 12 months searching for a much better name without knowing it. For me, Web services is the whole Java/JavaScript confusion all over again.

Having realized a couple of months ago that I was indeed on a mission to find a new name for Web services, my standpoint became a little different and I began trying to tease suggestions out of people without them knowing what I was up to. I wanted an honest and simple name, something that summed up the whole evolution without using the words "Web" or "service."

I am proud to say I think I have found something that I believe is better than what we are struggling with at the moment. Sadly I can't take the credit; that would fall to Mel Stockwell from IONA. In a lively e-mail exchange he uttered a phrase that I simply had to pounce on: "middleware for the masses."

You may disagree with it, but I think Mel has stumbled onto something *big* here and until I hear something better, I'll be using this to describe solutions that are open, available, and flexible (aka Web services).

• • •

Speaking of marketing...our "friends" at Microsoft are at it again. In fact, this month they have incensed me so much that this editorial is coming to you via StarOffice, not Word! (I have to admit, I'm pleasantly surprised at how good version 6.0 is.) Late last month I was sent a URL to a site that, at first glance, you would be forgiven for thinking was a third-party developer's

–continued on page 88

AUTHOR BIO

Alan Williamson is editor-in-chief of Java Developer's Journal. During the day he holds the post of chief technical officer at n-ary (consulting) Ltd, one of the first companies in the UK to specialize in Java at the server side. Rumor has it he welcomes all suggestions and comments!

alan@sys-con.com

togethersoft

www.togethersoft.com

KBrowser's Performance?



Jason Briggs' review of KBrowser (Vol. 6, issue 11) was interesting and useful. Thanks for highlighting it.

One area he didn't cover was KBrowser's performance. I'd like to know his impression of how it performed, not only for WAP content but also for simple HTML content as well. I'm

also interested in how he thinks KBrowser will perform on J2ME-enabled wireless handsets, now and in the future.

Ron Stein
ron@nazomi.com

I didn't include performance, because I don't have a real phone to run it on, unfortunately. Up until the last month or so there haven't been any MIDP-capable phones in the UK except for Nokia's PDA phone combination (which has a 55MHz processor and runs PersonalJava out of the box – not really a useful benchmarking system in this case).

Motorola has just released a MIDP phone here, but I've yet to get my hands on it.

Performance on my laptop, however, was fine, but not really an indicator of how it will actually run on a phone – your guess is as good as mine. However, gut feeling (from running MIDP apps on a Palm and on the aforementioned Nokia) says unless they're doing something exceptionally illogical, not that likely I think, then performance will be good enough so it will all finally depend on the network speed.

I couldn't run it on HTML content, because the evaluation edition they sent me was WAP only.



Jason Briggs
jasonbriggs
@sys-con.com

Is New Technology Always Good?

I agree with Alan Williamson's comments in his editorial, "<Web Services & XML>" (Vol. 6, issue 11).

For Web services, the battle is not defining how to create and/or access Web services, which is

where we're currently at. The difficulty is in changing Web usage. Folks are just getting comfortable with URLs, and while they benefit from not depending on a provider for a service, it also gives them some stability. I think it makes sense when architecting or building new services (which don't have issues with stability or familiarity), but users are another ball game.

As far as XML usage, it seems that we tend to throw a new technology at everything, mostly because we've learned something new and want to make it useful. But it can degrade and impede a system's internal communication.

Perhaps Web services will become more important if, as I predict, more robust user interfaces are doing the interacting with them, rather than users in a Web browser.

Vince Marco
vince.marco@iname.com



Enough Is Enough!

When are Bill Baloglu and Bill Palmieri (Career Opportunities column) going to seek help for the bottled-up anger they have toward anyone under the age of 50! Every article they write is essentially about how dumb the rest of us are (the under-50 crowd) and how smart they are.

Why do they feel the need to write article after article about how apparently no one is qualified to call themselves a Java engineer unless they have 20 years of experience? I personally don't consider myself a Java engineer, but I have worked with a few that I would consider that level and none of them were older than 30. I've also worked with developers with 15–20 years' experience who I could code around the day I stepped out of college.

Am I saying that all employees over the age of 50 are over their heads in the technology field? Absolutely not. But please admit that there are some people in their thirties and even their twenties who are intelligent and experienced and could do the job! If a person can do the job, then he or she will get the assignment.

Richard Dean
richarddean@yahoo.com

After a thorough search of all our articles, I couldn't find a single reference to age.

Our January article (Vol. 6, issue 1) dealt with the "know" engineer and the "understand" engineer – no mention of age there.



Our February article (Vol. 6, issue 2) talked about the B2B marketplace – no mention of age there.

March (Vol. 6, issue 3) talked about money levels for senior, mid-level, and junior engineers – no mention of age there.

May (Vol. 6, issue 5) discussed how to write an effective résumé – no mention of age there.

I think you get the picture.

There is a reason why we use the term senior. It has to do with the length of time of actual work experience as well as the technological expertise an engineer brings to his or her job. The senior engineers we work with are not over 50, or any specific age. However, they do possess the skills and years of experience we talk about.

Until then, we wish you the best and keep sending us your e-mails.

Billy Palmieri
billy@objectfocus.com

Important Issues

I just read Charles Arehart's excellent article "The Many Sides of J2EE Development" (Vol. 6, issue 11). I've learned a lot from this article and his previous one, "Making the Move to J2EE" (Vol. 6, issue 9). Mr. Arehart presents important issues in an excellent manner. I'm beginning to see that servlets are a more important part of the J2EE spec than I had previously imagined.

Frank Staheli
BYU

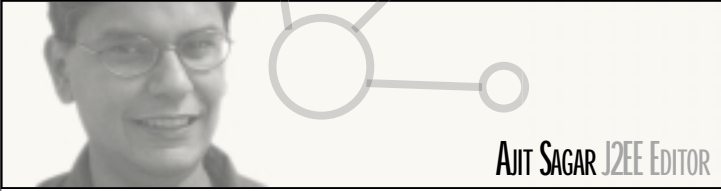
Don't Use Expensive EJBs!

It was such a joy to read "J2EE Without EJBs?" by Vince Bonfanti (Vol. 6, issue 11).

Many people believe that J2EE has to be implemented using EJBs. Some clients spend millions just to have a J2EE application. It's great to have Vince Bonfanti clear the confusion and wake up companies so they can save millions on a

J2EE application by simply using Java, servlets, JSPs, and JDBC instead of expensive EJBs!

Leo F. Smith
leo_smith@canada.com



AJIT SAGAR J2EE EDITOR

Java for the New Economy

Welcome to 2002 J2EE. The year 2001 has been a learning experience for all of us in the Java technology universe. The lesson has been a painful one – focus on the business problem and apply technology to ensure the right ROI. About a year ago, several folks were riding out the fantasy of paper money; options would change their entire lifestyle. They were going to take a year off and get back to work when they felt like it. Their requirements for cars and houses had taken on a whole new dimension. They were asking the world to excuse them while they kissed the sky. Then the bubble burst, and we're all back to basics again; living a fairly good life, but with no drastic changes. Last year, in anticipation of this revolution, the industry introduced several products to support it. These products are now on the shelves of discount stores.

For the past few years there has been the dream of a global enterprise driven by technology. That dream is still alive, but it has been tempered a bit by reality. The reality check has impacted all technology platforms, and Java is no exception. In retrospect, the partitioning of the Java platform into three editions has made it much more feasible for developers and architects to justify using the right combination of Java APIs for the right problem. Selecting the appropriate mix of Java features out of a monolithic platform would have been a formidable task in the current economic climate. In addition, the efforts of Sun, the IDE vendors, and the application server vendors have helped architects define how Java can be applied in their particular domain. In that sense, the Pet Store Demo was much needed in the Java industry.

Let's look at some of the facets of the J2EE platform that have been affected. The core of J2EE is the EJB object model. As I mentioned in my last editorial (*JDJ*, Vol. 6, issue 12), the J2EE application server market will suffer as a result of companies

reevaluating their problem domain's requirements. Many applications don't require the full power of a heavy and complex middle tier. The Web layer, which uses the Web application server (as opposed to the J2EE application server) and goes through homegrown data access modules to the data source, is becoming an attractive alternative. JSPs and servlets have already gained a lot of momentum. Container providers such as Tomcat and iPlanet Server are becoming viable alternatives to J2EE application servers. Not to say that the EJB market has died, only that it's being evaluated more judiciously to make sure it's the right solution for the problem.

ERP connectivity is a large piece of the puzzle and the focus of applications has shifted from global information exchange to interaction with traditional EIS sources. For that reason, the Java community is looking at JCA to provide developers with the means to get to a variety of ERP sources. At the same time, ERPs are exposing their core functionality via Java APIs and Web services. In 2002, we should see a maturing of the JCA and alternative connectivity mechanisms.

For the last few years Java internationalization has been a crucial part of all applications. Though this continues to be the case, the requirements for internationalized software will probably change. Consequently, leveraging Java APIs for internationalization will no longer be the focus. For example, last year at my company we were looking at a requirement for supporting an interface in two languages for the same desktop. This was for a site that would manage an auction for two different clients. Since public marketplaces have fallen out of favor, such requirements are rare today.

Finally, another example of basic requirements that have changed are those associated with security and user management. Self-registration and Web access to corporate applications are not as common now as they were earlier. Therefore, the

Java for the New Economy

For the past few years there has been the dream of a global enterprise driven by technology. That dream is still alive, but has been tempered a bit by reality.
by Ajit Sagar

8

J2EE Testing

An overview of the CTS for J2EE component developers
by Jonathan Maron

12

The Key to Superior EJB Design

Most of the EJB design practices created so far are aimed at improving the overall performance of EJB-based applications.
by Boris Lublinsky

20

J2EE Application Security

Container versus application-managed security
by Timothy Fisher

28

Core J2EE Patterns

Presentation-tier patterns and refactoring and how they relate to one another
by Dan Malks

32

Journeyman J2EE

A quick JavaScript lesson
by Charles Arehart

38

requirements for firewalls and single sign-on have changed a lot for enterprise applications. The requirements for integrating applications into a common security layer for the intranet are becoming more common than the ones for the Internet. Again this impacts which areas of Java will be leveraged to solve specific problems.

The good news is that the J2EE platform continues to evolve to meet the needs of the new economy and continues to be the platform of choice for enterprise applications. ●

ajit@sys-con.com

AUTHOR BIO

Ajit Sagar is the J2EE editor of *JDJ* and the founding editor and editor-in-chief of *XML-Journal*. A lead architect with Innovatem, based in Dallas, he's well versed in Java, Web, and XML technologies.

hp
www.hp.com

WHAT IS THE CURRENT VERSION OF THE JAVA 2 ENTERPRISE EDITION?

The Java platform versioning is quite confusing. This is even more true since Sun split the original platform into three editions. Although it was a good step for decoupling application development, it has led to more effort in terms of version management.

To answer the question, Sun released Release 1.3 of the enterprise edition (J2EE) on September 24, 2001. This is different from Release 1.3 of the JSE (Java Standard Edition), which was released more than a year ago. Currently the latest version of J2SE is 1.3.1 and the 1.4 version is in the beta release stage.

There's a dependency between the editions. J2EE uses the J2SE libraries, therefore the J2EE released versions need to be compatible with the latest J2SE versions. If you get to the granular APIs, they have their own versioning. For example, the J2EE Connector Architecture is in Release 1.0 and the Servlet API is in Release 2.3

Java 2, the current release of the Java platform, is the umbrella that encompasses the latest versions of all the API editions mentioned above. So the J2ME, the J2SE, and the J2EE APIs are all a part of the Java 2 platform.

CAN AN RMI SERVER RUN ON A JAVA CLIENT?

Yes, it can. The main components of an RMI-based application are an RMI client, RMI server, and RMI registry. The RMI server can run on a machine that hosts a VM that supports the RMI library. Support for RMI is required of all Java VMs.

In the traditional client/server scenario, a Java client application connects to a Java server application; the Java server application does the bulk of data processing and business logic on the server side. If you aren't connecting via the HTTP layer, i.e., your connectivity isn't based on access through a Web server via servlets/JSPs, you'll probably use RMI for connectivity. Typically, the RMI server will run as a part of your Java server application. That is, the implementation of the RMI-based interfaces for your application will be on the Java server application.

However, this isn't necessarily true. If you wanted to use RMI for some processing on the Java client and return the results to the Java client, you can achieve that also. In this case the RMI server will run on the Java client and the RMI client will run on the Java server. An example of this is when you want to push data to the Java client and invoke a method on the client to process that data on the Java client-side after receiving it from the Java server.

IS EJB A SPECIFICATION OR AN ACTUAL JAVA COMPONENT?

Both. The EJB specification defines the design and implementation for Enterprise JavaBeans, which are Java components. An interesting point to note in documents covering EJBs is that both the singular and plural word form is used to refer to EJBs. For example, Enterprise JavaBeans is a specification. Enterprise JavaBeans are Java components.

IS EJB OBJECT A REMOTE OBJECT?

Actually EJBObject (javax.ejb.EJBObject) is an interface that extends java.rmi.Remote interface. Note that an "EJBObject" interface and an "EJB Object" class (notice the space) are two different entities. An "EJB Object" class is an actual class that is autogenerated by the application server. The "EJB Object" object is a delegator object that exposes all the methods of your actual EJB (entity/session/message-driven) implementation.

When developing an EJB component, the first thing you need to do is define an interface (e.g., MyInterface) that extends the javax.ejb.EJBObject interface. In doing so, you'll add your business logic methods to the methods already defined in the EJBObject interface. These methods are used by the container to manipulate your EJB component – methods to find the bean, get a reference to the bean, etc. During deployment the container will generate a class that provides an implementation for all the methods for MyInterface, including delegator classes for the methods that you define. This class is your "EJB Object" class implementation of your "MyInterface" interface. ☘

J2EE ROADMAP

The Java 2 Platform, Enterprise Edition defines the APIs for building enterprise-level applications.

J2SE	v. 1.2
Enterprise JavaBeans API	v. 1.1
Java Servlets	v. 2.2
JavaServer Pages Technology	v. 1.1
JDBC Standard Extension	v. 2.0
Java Naming and Directory Interface API	v. 1.2
RMI/IIOP	v. 1.0
Java Transaction API ..v.	1.0
JavaMail API	v. 1.1
Java Messaging Service	v. 1.0

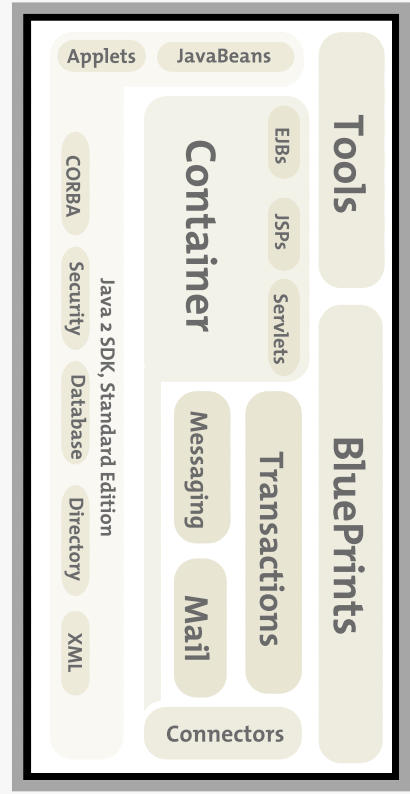
Useful URLs:

Java 2 Platform, Enterprise Edition
www.java.sun.com/j2ee/

J2EE Blueprints
www.java.sun.com/j2ee/blueprints

J2EE Technology Center
<http://developer.java.sun.com/developer/products/j2ee/>

J2EE Tutorial
<http://java.sun.com/j2ee/tutorial/>



sharp
www.sharp.com

What's It to You?



An overview of the CTS for J2EE component developers



WRITTEN BY
JONATHAN MARON

There are two fundamental attributes developers look for in a given technology to ease their development tasks: extensive functionality and code reusability.

Look at the success of the Unix operating system. A fundamental part of its success is that it can be written in a machine-independent language and ported to multiple hardware platforms. Thus it provides robust functionality that can be reused on multiple platforms. However, despite the extensive similarity between most of the popular Unix flavors, there's no guarantee that an application written for one Unix platform will be portable to another without making significant changes to the application's code. Thus, it falls short as far as code reuse is concerned.

A key design goal for the Java platform from its inception was to address this code portability issue. The "Write Once, Run Anywhere" mission of Java is paramount to its success. A developer no longer needs to code to a specific hardware platform; rather, the code is

interpreted and run in a virtual machine that shields the developer from the underlying hardware and operating system.

The Java 2, Enterprise Edition (J2EE) platform extends this portability to a set of enterprise components that run in a J2EE server. A developer coding to these APIs now not only has the assurance that the code will run on any platform supporting a JVM, but that the components will be portable across any container supporting the J2EE APIs.

Plus, J2EE component developers can get further assurance that this is true through the J2EE Compatibility Test Suite (CTS). The CTS was created to ascertain the portability of developers' enterprise components by testing whether an application server meets each individual requirement set forth in the J2EE specifications.

This article presents the types of tests addressed by the J2EE CTS, the future of the CTS, and further explores the benefits as well as the limitations of the CTS for the J2EE developer.

The CTS

The J2EE CTS includes the tests and tools necessary to ensure that a J2EE-based application server works as established in the J2EE specifications. It consists of several thousand tests that explore whether all the required APIs are present, the necessary services are provided, and the contracts between the required components work as expressed in the specifications. Once the CTS is successfully completed by a given vendor, that vendor's application server is

considered J2EE-compatible and receives the J2EE brand. Being J2EE-compatible implies that an application server meets the requirements that guarantee the stability of a J2EE application component in any J2EE environment.

The CTS tools consist of a GUI console for executing and monitoring the tests (see Figure 1) and an associated testing harness. The testing harness consists of a portability component that allows a vendor to plug in a set of classes that govern the deployment and removal of the test applications. In addition, the harness defers to vendor-specific classes for the creation of URLs that access the Web-tier components of a test. Finally, vendor-supplied porting kit classes govern JNDI lookups of deployed test application components.

There are four major server components in a J2EE server environment: application components, containers (e.g., EJBs, application clients, and servlet/JSP), resource manager drivers (e.g., JDBC), and one or more databases. The J2EE specifications define a set of standard services that each of these components must support. Application components access these services through APIs provided by the J2EE containers. The CTS thoroughly exercises each of these server elements via direct API tests or via end-to-end integration tests that may repeat the execution of a given test in each of the server's containers.

The J2EE CTS ensures that the application servers abide by the requirements set forth in the constituent set of APIs as shown in Table 1.

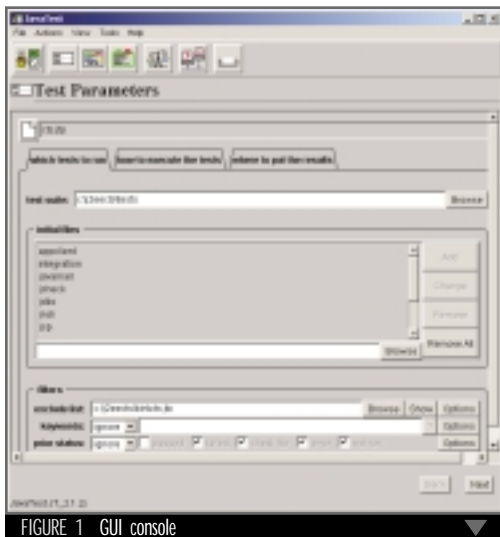


FIGURE 1 GUI console

mongoos

www.mongoos.com

infrag
www.infrag

gistics
gistics.com

Types of Tests

There are three types of compatibility tests: signature, API, and end-to-end or integration. *Signature tests* are executed across all the J2EE technologies to ensure that the correct APIs are available to the application components running in the server. *API tests* are executed for each specific technology to ascertain whether the application server meets the requirements of the associated specification (see Table 2). Finally, the *integration tests* examine the interaction between various application components to guarantee that request invocations are propagated appropriately (e.g., resource references are obtained successfully, transactions are propagated).

The API and integration tests are executed in a variety of execution contexts to ensure support for the required application components; a given test is repeated from an application client, an EJB, a servlet, and a JSP.

J2EE 1.3 CTS

In the recent release of the J2EE specifications (version 1.3), several specifications are either modified significantly or are added to the platform. These platform modifications resulted in added complexity in the compatibility process. Therefore, the J2EE 1.3 CTS includes newer sets of tests to verify the

compatibility of J2EE 1.3 application servers, including:

Connectors

Connector tests verify that an application server has all the required pieces that allow any compliant resource adapter (RA) to be deployed and work correctly. A resource adapter is a system-level software driver that's used by a Java application to connect to an Enterprise Information System (EIS) such as CICS or IMS.

EJB 2.0

The EJB 2.0 specification contains some dramatic changes from earlier specification versions. The most notable change is the new container-managed persistence (CMP) model, the implementation of which relies on some new technologies such as the new EJB query language.

J2EE CTS 1.3 tests are required to check the conformance of an application server with the EJB query language requirements of the EJB 2.0 specification. These tests focus on the runtime execution of the query and not explicitly on the language syntax. To be able to accomplish the majority of the queries needed in the test cases (derived from the extensive number of queries defined in the specs), an abstract persistence schema had to be implemented. This schema provides

the necessary relationships, dependent objects, and entity bean classes that provide the necessary support for those queries.

Other changes include a new bean type called message-driven beans (MDBs), home methods independent of a specific EJB instance, and requirements for security and transactional interoperability. In addition, new Local and LocalHome interfaces were added to support the CMP 2.0 model. The interoperability tests verify that security propagation, transaction propagation, remote method invocation, and naming services can access all work and interoperate across multiple J2EE-compliant platforms.

Java Message Service (JMS)

The tests in this area verify compliance with the JMS functionality expressed in the EJB 2.0 specification, the J2EE platform specification version 1.3, and the JMS specification version 1.0.2. These tests are derived from the requirements to provide message-driven beans with JMS access from the various J2EE containers, and the capability to perform JMS operations in a distributed transaction. The appropriate JMS API tests are also included.

JDBC 2.0

The new JDBC tests added in CTS 1.3 verify that an application server implements the required JDBC 2.0 extensions expressed in the J2EE 1.3 specifications.

Java Transaction API

The new transaction-related tests in CTS 1.3 check compliance with the new CMP model introduced in the EJB 2.0 specifications. Specifically, tests will be included to validate access to multiple databases within the context of a transaction.

Security

Security testing in CTS 1.3 covers the compatibility requirements in multiple specifications, including J2EE 1.3, EJB 2.0, Servlet 2.3, and JMS 1.0.2 specifications. Security coverage for the J2EE connector architecture is not included in these tests.

JSP 1.2/Servlet 2.3

The JSP and servlet tests cover the new functionality related in the new specifications such as the new tag library extensions, JSP precompilation, and servlet filtering.

CTS—What It Is (and Isn't)

Implementing a J2EE API requires reading and interpreting the core API

TECHNOLOGY	J2EE 1.2	J2EE 1.3
Enterprise JavaBeans (EJB)	1.1	2.0
Java Servlets	2.2	2.3
JavaServer Pages (JSP)	1.1	1.2
Java Transaction API (JTA)	1.0	1.0
RMI/IIOP	1.0	1.0
JDBC	2.0	2.0 Extensions
Java Naming and Directory Interface (JNDI)	1.2	N/A
JavaMail	1.2	1.2
Java Message Service (JMS)	NA	1.0
Java API for XML Processing (JAXP)	NA	1.1
J2EE Connector Architecture	NA	1.0
Java Authentication & Authorization Service (JAAS)	NA	1.0

TABLE 1 J2EE CTS for 1.2 and 1.3

API	TESTS
Enterprise Java Beans	EJB deployment, bean behavior, security, and transactions
Servlets	Servlet methods, servlet context interactions
JSP	JSP security, JSP implicit object access, servlet context interactions
JTA	Transactional propagation, user transaction access
RMI/IIOP	Support for remote exceptions, primitive array propagation, transmittal of arrays of remotes, etc
JDBC	Full JDBC 2.0 API support
JNDI	Full JNDI 1.2 Support
JavaMail	Transmittal of messages, support for utility methods, MIME type support, multipart message support
JMS	Message-driven beans, access to JMS provider

TABLE 2 API Tests

sybase

www.sybase.com

specifications. Although the specifications are for the most part clearly written, there are a number of ambiguities. These ambiguities, which may not become apparent until multiple implementations exist, often lead to varying interpretations on the part of the J2EE licensees. A process is needed that not only validates the various implementations, but also the fundamental interpretations that formed the basis of those implementations.

The tests included in both the current and upcoming versions of the CTS provide a concrete representation of the requirements listed in the respective J2EE specifications. The application server vendors use these tests to validate their understanding of the specifications. A licensee is much more apt to successfully implement a given technology when presented with concrete examples showing how components interact with the given API. In addition, the CTS provides a forum for the vendors to resolve specification ambiguities.

One of the major challenges for the developers creating the CTS tests is that at their inception, no actual implementations existed other than Sun Microsystems' J2EE Reference Implementation (RI). Therefore, the tests are executed and validated against the RI.

However, the RI may have some inherent specification assumptions and interpretations that are erroneous, leading to tests that are unsound. If a vendor disagrees with a given test, a dialogue is initiated with Sun's CTS team to resolve the issue. This process ultimately resolves whether the test represents a misinterpretation of the specification, is flawed and could never be passed, or whether the vendor's implementation does not support the requirement represented by the test. If a test is deemed unsound for any of these reasons, it's either patched (if removing it would compromise the compatibility process) or added to an exclusion list (i.e., a list of tests that are no longer executed by the CTS).

The process outlined above presents the J2EE component developer with numerous benefits, as well as some distinct disadvantages. Generally, a developer can regard the J2EE CTS process as a standardization effort that ultimately normalizes the behavior of his or her application components. For example, if multiple vendors pass the same set of tests that address the functions of the servlet context, a developer

can be confident that his or her use of the servlet context will manifest the same results, no matter which server is utilized.

However, the fact that certain tests have potentially been excluded implies that the current suite of tests may not have complete coverage of the J2EE APIs. Although vendors are obligated by their participation in the J2EE branding process to provide a fully compliant J2EE server, without a thorough and complete set of tests there's no guarantee that such compliance has been achieved.

In addition, certain APIs may not lend themselves to automated testing and may require manual intervention. For example, the Secure Sockets Layer (SSL) requirements of JavaServer Pages (JSP) can't be tested without a fully SSL-compliant browser invoking the requests.

It's also important to note that the CTS was not created to address the performance, robustness, ease-of-use, or scalability of a given J2EE application server. The CTS is a methodical examination of compliance with the requirements of the J2EE specifications. It makes no attempt to time the tests or run the tests under a significant client load (performance metrics will be addressed by JSR 4 of the JCP – the ECPeB Benchmark Specification). To make a judicious choice of an application server, a developer should not only ascertain whether the server is J2EE-compliant, but also how the server performs, how easily it's configured, and whether it's fault tolerant.

Moving forward, the developers of the CTS have a substantial challenge to keep up with the breadth of the J2EE platform. The overwhelming number of new and modified requirements will translate to a need to substantially increase the number and scope of the associated tests. Ensuring that all the requirements are addressed and sufficiently tested will present a major effort.

Finally, developers need to be aware that the J2EE CTS is not backwards-compatible, although most of the API specifications under the J2EE umbrella do require old application components to continue to operate; in other words, an application server that has any J2EE 1.3 component support can no longer be considered J2EE 1.2-compatible and must proceed on the path to achieving J2EE 1.3 compatibility. To developers, this means they may be confronted with releases of application server products that no longer display the J2EE brand

even if a previous release was officially compatible. In some cases, J2EE 1.3 platform implementations may be released with a "beta" or "early release" label. Generally, developers should rest assured that the vendors are continuing to support the J2EE APIs and will release compatible versions as soon as they can support the large number of new requirements in the J2EE specifications.

Summary

The compatibility testing process encompassed by the J2EE CTS provides a comprehensive test of an application server based on the requirements and design guidelines established in the J2EE specifications. The CTS also attempts to provide full coverage of the strict requirements of each J2EE-related specification.

For developers, the CTS provides a significant level of assurance that coding to the J2EE APIs will yield an application that can be deployed on any certified J2EE application server, so developers no longer need to learn vendor-specific proprietary APIs. This also means that developers have a much more portable skillset.

The CTS also holds significance for corporate IT managers, even though the importance of portable code for people in these positions is probably negligible. As a corporate IT manager, if you purchase and deploy a certified J2EE application server (that is, one that has passed the CTS), your "hiring pool" – the candidates qualified to write and deploy your applications – increases exponentially. Instead of spending time and resources looking for developers with skills specific to one vendor, you need only look on a candidate's résumé for experience coding to J2EE standards. In addition, the decision as to which application server will host the resultant application can be made parallel to the development effort; the CTS goes a long way toward ensuring that the application will deploy successfully and run on a certified J2EE application server.

The CTS isn't a guarantee of any kind, nor should it be the only criteria used to select an application server. Rather, the CTS is an equalizer; it provides application server vendors with a platform from which they can differentiate their products through functionality, implementations, and so on.

In other words, the CTS isn't the answer, but it's an important part of the whole equation. ☛

jonathan_maron@hp.com

AUTHOR BIO

Jonathan Maron is a distinguished engineer with Hewlett-Packard's middleware division. He was the lead on HP Bluestone's J2EE 1.2 compatibility effort, and also principal architect of the Bluestone EJB 1.1 server. Jonathan is a member of the JavaSoft EJB and J2EE expert groups.

object design

www.objectdesign.com

The Key to Superior EJB Design

Decrease network traffic in *EJB* implementations

Over the past several years
EJB technology has
entered the software
development
mainstream.
This new level
of recognition
and greater
popularity
brings an
increase in design
activities in the EJB
space, such as best
practices and design patterns.

Written by Boris Lublinsky



J2ME



J2SE



J2EE



Home



Most of the EJB design practices created so far are aimed at improving the overall performance of EJB-based applications. It turns out that the majority of these practices were taken directly from object-oriented development (OO) and moved to the realm of EJB design, without consideration for the specifics of EJBs. This article emphasizes these specifics and how they impact the design of EJBs and EJB-based applications.

What's So Special About EJBs?

EJB technology was introduced as a distributed components technology. The key to understanding it lies in the meaning of the words *distributed* and *components*. Let's start with distributed, then examine components.

Distributed Aspects

EJBs are accessed through the Java Remote Method Invocation (RMI), regardless of whether they're local or remote to the client. Although some of the application server implementations (e.g., WebSphere) optimize local communications to make them faster, most EJB communications are still network-based. Although the distributed aspect of communications is transparent to the user in actual method invocations, it has a profound effect on execution performance. The situation is further complicated by the fact that actual communication with the bean is based on interception (see Figure 1) and is implemented in two steps:

1. A request for the bean's method of execution is first sent to the container in which the bean resides.
2. The container fulfills the required intermediate steps (security, transactions, etc.) and then forwards the request to the bean.

For the method on the EJB to be invoked, the remote reference to the home interface must be obtained. This is usually done through an additional network call to the Java Naming and Directory Interface (JNDI). The home interface can then be used to get the actual EJB reference. These operations introduce additional network calls (see Figure 2).

To summarize, the execution method on the EJB is an expensive network process. Thus having low granularity methods on the EJB typically lead to poor performance of the overall system.

The introduction of local interfaces in EJB 2.0 is one attempt to improve overall performance. Local interfaces provide a way for beans in the same container to interact more efficiently – calls to methods in the local interface don't involve RMI. Although the local interfaces represent EJBs in the same address space and don't use distributed communications (e.g., no RMI between colocated beans), the container is still involved in every interaction to provide the required intermediary steps. In addition, even in the case of a local interface, a networking call to the JNDI is required for the client to obtain a reference to the local home interface, through which a reference to the local interface can be resolved. In reality, the specification doesn't define how vendors must implement local interfaces since they're only logical constructs and may not have the equivalent software counterparts. Additional delays can still be present in local communications.

The only effective way to improve the overall performance of EJB-based applications is to minimize the amount of method invocations, making the communications overhead negligible compared with the execution time. This can be achieved only by implementing coarse-grained methods.



J2ME



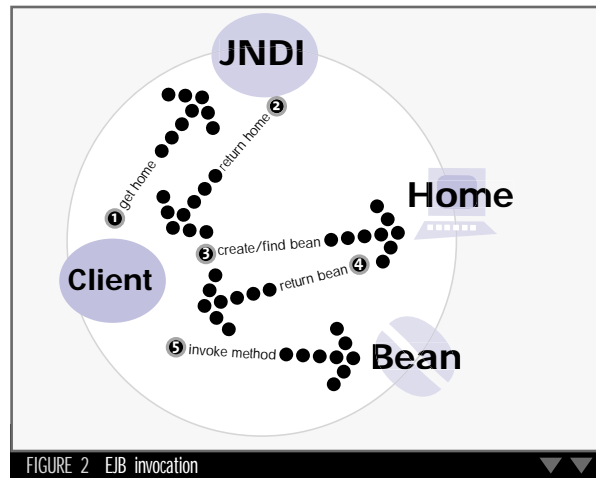
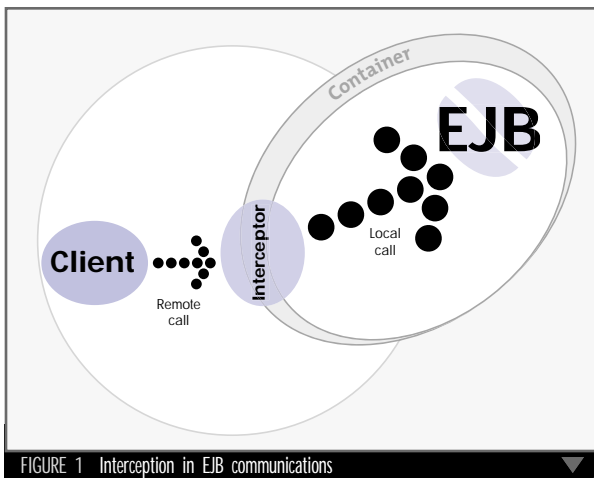
J2SE



J2EE



Home



Component Aspects

To define the component characteristics of EJBs we have to first define what components are. Although component-based development (CBD) has been around for at least 10 years, they're still not clearly defined. In general, components are for composition. Composition enables the reuse of pre-fabricated "things" (components) by rearranging them into ever-new and changing composites. Beyond this observation there's a lack of consensus on the definition of a component within the software industry. Microsoft has even invented the Component Object Model (COM), thus implying in the name some relationship between components and objects.

The Object Management Group (OMG) has defined distributed objects and built distributed components on top of them, leading people to think that components are tightly linked with objects. Many people assume that components are nothing more than super objects – a huge misconception.

- Must be identifiable, meaning it can be addressed by another component, possibly via a network.
- Should be treated as a whole so that it's not necessary to worry about all its pieces. This requires that components can be individually designed, developed, and deployed.
- Should separate its interface from the implementation used to support it. A component might be thought of as a "black box" implementation of the business construct with a well-defined interface.
- Component-based development (CBD) is not object-oriented development. This means that CBD does not necessarily require OO development. CBD can be implemented with equal success in both OO and procedural languages. CBD is merely a way of decomposing systems. It's a way to manage complexity better.

Most people consider the potential for reuse to be the main driving force for using a CBD approach. To be inde-



“

Components allow for the natural decomposition of a complex system into smaller chunks, which are usually much simpler and easier to manage

”

Components are a software implementation of business artifacts, intended to simplify the creation of business applications. Objects are software constructs, intended to simplify code creation; they're not necessarily related to the business content of an application.

Instead of trying to come up with a precise definition for components, we'll define the core concepts the industry is using as a "unified" description of a *software component*:

- A software implementation of a well-defined application (business) aspect.
- Should implement a collection of related functions or services; a relationship is determined by the analysis done from the perspective of intended usage. The component should provide a complete but not necessarily exhaustive set of functions.

pendently deployable a component has to be self-contained – separated from its environment and other components. Coupled with the requirement to implement well-defined application aspects, this provides the widest possibility for reuse.

Managing complexity is another major advantage of CBD. Components allow for the natural decomposition of a complex system into smaller chunks, which are usually much simpler and easier to manage. In addition to horizontal partitioning, introduced by layered architecture, the adoption of components introduces vertical partitioning.

The description of a component provided earlier does not specify the internal implementation of the component. This means that in principle, components can be implemented using lower granularity components (e.g., IBM's advanced components for WebSphere). This is similar to the system-analysis paradigm in which large systems are believed to con-

data direct

www.datadirect.com

sist of smaller systems, recursively, until the size of the system becomes manageable.

This recursive definition lets you think about components as a unifying concept for the software system as a whole as well as individually. The introduction of components also forces a multilevel design: the components and their internals. A compound component is made up of several components.

The following is a summary of the benefits of CBD:


- **Containment of complexity:** Using CBD allows for the natural decomposition of a system. First, create a high-level design of the components and their interfaces. Then focus your development project on one or a small number of components. This effectively allows for the reduction of scope and better risk management of every project. Besides, smaller and better-focused development teams are usually more productive.
- **Opportunity for massive parallel development:** Project boundaries defined around stable component definitions encourage parallel development in-house and via outsourcing. The outsourcing of maintenance may occur as well, since component providers may supply maintenance for their components.
- **“Black box” component implementation encourages flexibility:** A component that supports a well-defined interface can be substituted with another one that supports either the same interface or one derived from the original interface. This simplifies modifications to current behavior and enhances functionality.
- **Incremental testing:** Components facilitate unit testing and support progressive build testing.
- **Encapsulated components act as firewalls to change:** The ripple effect from change is much smaller, simplifying system maintenance.

JavaBean and EJB development, thus suggesting a strong correlation between the two distinct technologies.

One of the examples of such correlations are setter and getter methods, which are required by the JavaBean specification to access internal variables. Setter and getter methods were introduced by OO practitioners in order to provide access to encapsulated object variables and eliminate coupling between internal representation and external access. This practice was blindly moved into EJB development, after which time many additional patterns – most notably the Façade and Value Object patterns – were introduced to improve design performance, which was less than optimal to start with.

Experience has proven that using setter and getter methods in distributed systems is a bad habit. Further, one of the rules for distributed computing is the introduction of self-contained method signatures to minimize network traffic and improve overall performance, which setter and getter methods rarely embody. The main characteristics of a self-contained method signature is that it accepts all the variables required for the method execution and returns all the results of the execution. In other words, self-contained methods don't require additional methods for either setting required data or retrieving results. Furthermore, because components are an implementation of application (business) artifacts, the methods that they support are supposed to be meaningful business methods, which setters and getters rarely are.

Our point is that a single EJB must be a large granular piece of software that's internally composed of a potentially large number of Java classes. It has to represent meaningful business artifacts and support meaningful business methods. This is the only feasible way of creating high-performance EJB applications with reusable beans.



The implementation of EJB-based components dictates a new approach to the design of EJB-based systems

- **Greater consistency in usage:** Components impose a standard architecture for applications.

What Does This Mean for EJB Development?

It's now apparent from our distributed and component discussion that superior EJB design is very different from OO design. The problem is that this point was never fully carried across to developers, many of whom still consider EJB to be a Java class that adheres to the EJB interface specification. The individual deployment of EJBs is the only component characteristic supported and emphasized by the EJB environment.

Simply because of its name, *Enterprise JavaBeans*, EJB connotes a relationship with another popular technology from Sun Microsystems – JavaBeans. To make things worse and confuse people even more, many popular Java IDEs (e.g., JBuilder) use a single workspace or “bean tab” for both

Impact on Systems Design

The implementation of EJB-based components dictates a new approach to the design of EJB-based systems. It impacts the separation of responsibilities between session and entity beans as well as the design of the beans.

Entity beans are often introduced as persistent data components (enterprise beans) that know how to persist their own internal data to a durable storage area such as a database or legacy system. This definition reduces entity beans mostly to object/relational mapping and often leads to a design in which entity beans are used purely as a data access layer (we've even seen a comparison of entity beans with serializable Java objects, which serialize themselves into a database). In this approach entity beans become fairly small, with a one-to-one correspondence between an entity bean and a database table that leads to a very low granularity implementation.

This causes not only increased network communications, but also negatively impacts database communications due to

compuware

www.compuware.com



the increased usage of finder methods. The standard implementation of a finder method is a database query for the key value. As the number of entity beans of the same type grows, this lookup, which is a separate operation from the actual population of the entity bean, becomes more and more expensive.

Some implementations, for example, WebLogic, allow for the optimization of finder methods by combining them with the load. This alleviates the problem somewhat, but is not part of the standard. Also, as the variety of entity beans grows, the amount of finder method invocations also grows, making the overall application's performance even worse.

In addition, the granularity of entity beans has a profound effect on database design. Prior to the introduction of entity beans (and the componentization of software in general), database design was performed for the application as a whole. This usually led to a database design with a strong emphasis on enforcing data relationships by supporting entity relationships and multiple constraints. With the introduction of entity beans (e.g., components) the situation has to change. Because entity beans are reusable, individually deployable components, the only thing a database can enforce is that the relationships within the data are supported by the individual components (beans). Introducing relationships in data that's supported by multiple entity beans will break the beans' autonomy, so it doesn't seem to be a feasible solution.

The relationship between the data of multiple entity beans must be implemented on a higher level by the session beans as part of the internal business-process definition. The lower the entity beans' granularity, the less relationships can be enforced in the database and the greater the programming effort that's required to support them.

The last thing to consider here is the fact that business rules that govern enterprise processing can be divided into two broad categories:

- **Accessing data:** These rules govern how data has to be stored in the database, operations that can be done with this data, and possible constraints. These rules are usually part of the business artifact and tend to be very stable and applicable for multiple implementations both within and between enterprises, and to provide a high potential for reuse.
- **Processing data:** These rules govern business processes within the enterprise. They define both the conditions and the sequence of the components' execution. They tend to change fairly frequently and are rarely reusable.

Entity beans must incorporate two major things: persistent (enterprise) data and business rules that are associated with the processing of this data. Ideally, entity beans should be viewed as an implementation of reusable business artifacts and adhere to the following rules:

- Have large granularity, which usually means they should contain multiple Java classes and support multiple database tables.

- Be associated with a certain amount of persistent data, typically multiple database tables, one of which should define the primary key for the whole bean.
- Support meaningful business methods and encapsulate business rules to access the data.

A session bean should represent the work being performed for the client code that's calling it. Session beans are business-process components that implement business rules for processing data.

Business processes implemented by session beans within the EJB environment should define business and corresponding database transactions. It's not advisable to use a client's transactions in the EJB environment due to potential problems with the long-running transactions that can cause database lockup. Entity beans that participate in the transaction are effectively transactional resources due to their stateful nature. In reality, however, application server vendors don't treat them as such and basically "clone" entity beans when more than one user wants to access the same information. They rely on the underlying database to lock and resolve access appropriately. Although this approach greatly improves performance, it provides the potential for database lockup.

At the beginning of a transaction the container invokes a load method on the entity bean that's performing the database read, thus acquiring read lock on the set of tables. At this point another clone of the same bean can acquire the same data and obtain another read lock. After that first transaction has ended, the container invokes a save method on the first bean that tries to write data back to the database. The database would attempt to promote the lock to the write operation, but would not be able to because there's another read lock for the same data. As a result a database deadlock would occur.

The severity of this situation can vary, depending on the locking mechanism of the database in use and the duration of the transaction. Either way, it's not a desirable response.

Summary

Our main stipulation in this article is that EJB design is very different from OO design and it's impossible to blindly apply OO design principles to EJBs.

A simple example is designing for reuse. In OO systems the main driver is to reuse code constructs, and the best results can be achieved by creating objects of very low granularity. In component-based development and thus EJB development, the main driver is to create reusable business artifacts, thus components must be of fairly large granularity.

The creation of coarse EJB components that consist of multiple Java classes will eliminate much of the network traffic occurring in today's EJB implementations. It will also allow for two levels of reuse: traditional OO reuse on the Java classes level that provides a component's internal functionality, and the component's reuse on the EJB level.

Acknowledgment

Special thanks to Michael Farrell Jr. and Tung Mansfield for their contributions to this article. ☪

Author Bio

Boris Lublinsky, regional director of technology at Inventa Technologies, oversees engagements in EAI and B2B integration and component-based development of large-scale Web applications. He has over 20 years of experience in software engineering and technical architecture.

blublinsky@hotmail.com



sitraka

www.sitraka.com

J2EE Application Security



WRITTEN BY
TIMOTHY FISHER

When designing Web-based applications, security is a critical component. Before the advent of J2EE, to implement a secure distributed application you had to code all of the security directly into the application.

Container vs application-managed security

J2EE introduced a powerful security infrastructure for applications that greatly assists developers and enterprises in securing their applications. When used properly, this infrastructure takes much of the burden of securing the application off of the developers, leaving them free to concentrate on implementing business logic.

The J2EE container-security services primarily address the security requirements of authentication and authorization. *Authentication* is the mechanism by which callers and service providers prove to each other that they are acting on behalf of specific users or systems. *Authorization* mechanisms provide control over what resources an identified user or system has access to. In simple terms, authentication provides the answer to “Who are you?” And authorization provides the answer to “What can you access?”

Authentication

The J2EE model supports several methods for authenticating users. These are basic, digest, form-based, and certificate-based authentications. In basic authentication, the Web server prompts the user for a user name and password, which is then transmitted to the server. Unless an SSL session has been established, this information is sent in the clear, so it's not very secure. Digest authentication improves the security a bit by sending a digest of the user name and password along with some session-specific information to the server instead of transmitting the clear text password.

Both of these methods result in a standard dialog being presented to the user for entering user name and password.

Form-based authentication allows the developer to create a custom log-in page using a form. Like basic authentication, the user name and password are sent in the clear, unless an SSL session has been established.

Finally, the most secure method of authentication is the certificate-based method. In this method, both client and server use X.509 certificates to prove their

identities. This authentication always occurs over an SSL-protected channel.

The Web component deployment descriptor specifies which resources are protected, thus requiring user authentication. The actual step of authenticating the user is usually accomplished by looking the user up in a corporate directory or database.

After successfully proving a user's or service's identity, an authentication context is established. This allows the user or service to be authenticated to other entities – without repeating the authentication lookup step. A user may also delegate its authentication context to a component, allowing that component to call another component while impersonating the original caller.

The authentication mechanism is configured in the Web component deployment descriptor. Listing 1 shows an example of configuring digest authentication. Listing 2 shows an example of configuring form-based authentication. The error page specified in Listing 2 is a page presented to the user when authentication fails.

Authorization

J2EE employs a permissions-based authorization model. Each protected resource is listed in the deployment descriptor, along with a list of roles that are able to access the resource. These roles are mapped to specific users by the application deployer. Static pages, JSPs, and servlets are protected at the URL level and can be further protected down to GET or POST methods. Protection of EJBs can be specified down to specific-class methods. Specifying authorization information in the deployment descriptor is referred to as *declarative authorization*. Embedding authorization logic directly into an application is called *programmatic authorization*. The J2EE model supports both of these authorization models. Unless requirements demand it, declarative authorization is generally the preferred method.

Declarative Authorization

The authorization rules specified in the deployment descriptor are enforced by the J2EE container. This method of providing authorization frees the developer from worrying about implementing authorization, which is generally a good thing because it's an easy area for developers to introduce bugs that can lead to large security holes in the application. Applications that rely solely on this form of declarative authorization are sometimes referred to as security-unaware applications. The application code itself has no knowledge of the security that is wrapping it. Declarative authorization is a container-managed security service.

Listing 3 shows an example of applying declarative authorization to a servlet in the deployment descriptor. In this example, all servlets and other resources in the “restricted” directory will be protected. Only users assigned the role “AuthorizedUser” will be granted access to these resources. Listing 4 shows declarative authorization being applied to an EJB. This example applies protection to the UserInformation bean. Users assigned the “admin” role are granted rights to access all methods. Users assigned the “customer” role are granted the right to access the getDetails() method.

Programmatic Authorization

The J2EE model also provides support for extending the authorization model through programmatic authorization. There are four key methods developers can use. They are:

- isCallerInRole() and getCallerPrinciple() for use by EJB code
- isUserInRole() and getUserPrinciple() for use by Web components

These methods are typically used to provide finer-grain access control than what can be achieved through pure declarative authorization. Applications that employ programmatic authorization are referred to as security-aware applications. Programmatic security is also

rational

www.rational.com

referred to as application-managed security.

A good example of where programmatic authorization is necessary is in protecting access to user account objects. Consider an application that manages some type of user accounts.

Let's say the developer designates two roles: one for users and the other for administrators. The users should have access only to their own account. The administrators should have access to all accounts. There is no way to enforce this policy using declarative authorization alone. This stems from the fact that access rules specified in the deployment descriptor are class-based, not instance-based. Specific users or roles can't be granted access to specific instances of an object while denying them access to other instances of an object.

A solution is to use programmatic authorization. In the account object, the developer would use the `isUserInRole()`

method to determine if the user was an administrator. If so, the user would automatically be granted access to the account. If not, the developer would then use the method `getUserPrinciple()` and compare that to the account owner. If they are the same, access would be granted.

Implementing security inside each application greatly increases the chance of introducing a vulnerability that can be exposed by hackers. Programmatic authorization should be used only when the requirements can't be met through declarative authorization.

Confidentiality and Integrity

So far, I've covered the topics of authentication and authorization. The other important security requirement is message protection, specifically, employing the concepts of confidentiality and integrity.

Integrity ensures that a message hasn't been accidentally or intentionally modified. *Confidentiality* ensures the privacy of a message. Both of these services are provided through cryptographic techniques.

The J2EE security model doesn't provide a great deal of flexibility or support in this area. In the deployment descriptor, resources can be designated as requiring confidentiality. This permits them to be served only over SSL connections. Going beyond the pure J2EE security model, there's a great deal of support in Java for providing programmatic protection of messages.

Divided Responsibilities

One of the goals of the J2EE security model is to lessen the burden on the application developer for securing applications. To assist in achieving this goal, the J2EE model recognizes the three distinct roles played in bringing an application from concept to deployment. The roles are component provider, application assembler, and deployer.

Slight variations on these role names appear in various publications. The component provider is the developer, the person actually writing the Java code. The application assembler takes several components (beans, EJBs, JSPs, etc.) and assembles them into a complete application. Finally, the deployer is responsible for actually deploying the application into the production enterprise environment.

The component provider implements all programmatic security. In doing so, they define generic security role names for providing access control functionality. The application assembler defines logical roles, specifies which resources should be protected, and

which roles are required to access those resources. The deployer then is responsible for mapping the logical roles to the specific enterprise environment. It should be noted that one individual or team may play multiple roles in bringing an application to production. It's not unusual for the component provider and the application assembler roles to be performed by the same team.

Container-Managed Security

Implementing modern security mechanisms such as confidentiality, authentication, and authorization requires a great deal of expertise in the security domain. Most application developers don't have the expertise needed to successfully implement these security services. This makes creating applications that are security-aware all the more challenging.

A better idea is to rely on the application-server vendors to implement the tough security services – and have your application make use of these container-managed services. This eliminates the necessity of having a security development engineer on every Web application project.

Having said this, the requirements must be the ultimate driver for whether or not programmatic security should be used. Some requirements, such as the account example given in this article, can't be satisfied using container-managed security alone. It should be noted that this example and the methods provided by J2EE specifically focus on authorization. It's best to always rely on container-managed authentication. Embedding authentication logic into each application creates a hard to manage authentication policy that has a high potential for security vulnerabilities.

Another advantage of relying on container-managed services is that as authentication technology is upgraded within an enterprise, the applications don't have to be rewritten. Using programmatic authentication would require massive application rewrites if enterprise authentication technology were to change.

Relying predominantly on container-managed security also allows you to maintain a common security infrastructure as opposed to sprinkling security throughout all your Web applications. Container-managed security services allow for the centralization of security policies in a central repository. When using container-managed security, a change in security policy requires changes in application configuration, as opposed to coding changes in your applications. ☘

AUTHOR BIO

Timothy Fisher is a consultant for Spherion Technology in Detroit, and has been involved in information-security technology for nearly 10 years. He worked in information security at Motorola, Cyclone Commerce, and Pricewaterhouse Coopers.

Listing 1

```
<web-app>
<login-config>
  <auth-method>BASIC|DIGEST</auth-method>
  <realm-name>test</realm-name>
</login-config>
</web-app>
```

Listing 2

```
<web-app>
<login-config>
<auth-method>FORM</auth-method>
<form-login-config>
  <form-login-page>login.jsp</form-login-page>
  <form-error-page>error.jsp</form-error-page>
</form-login-config>
</login-config>
</web-app>
```

Listing 3

```
<web-app>
...
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
Secure Content
  </web-resource-name>
  <url-pattern>/restricted/*</ url-pattern>
</web-resource-collection>
  <auth-constraint>
    <role-name>AuthorizedUser</role-name>
  </auth-constraint>
</security-constraint>
...
<security-role>
  <description>
The role required to access restricted content
  </description>
  <role-name>AuthorizedUser</role-name>
</security-role>
</web-app>
```

Listing 4

```
<method-permission>
  <role-name>admin</role-name>
  <method>
    <ejb-name>UserInformation</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>

<method-permission>
  <role-name>customer</role-name>
  <method>
    <ejb-name>UserInformation</ejb-name>
    <method-name>getDetails</method-name>
  </method>
</method-permission>
```

Download the Code!
www.javadevelopersjournal.com

tim@securitydeveloper.com

quintessence systems

www.quintessence.com

Core J2EE Patterns

Presentation tier patterns and refactoring



WRITTEN BY
DAN MALKS

Patterns are expert solutions – recurring designs that have proven effective over time. This month’s article will provide you with a bit more detail on the subject.

Due to an unfortunate postproduction printing error, this article, which first appeared in the October issue of *JDJ* (Vol. 6, issue 10), was not published in its entirety. As we are committed to bringing you the best content possible, we couldn’t let this article slip by. Therefore we have included the complete article here, and please accept our apologies.

When applying patterns to the presentation tier, we address the following:

- *Pre- and postprocessing of a request and/or response*
- *Request handling, navigation, and dispatch*
- *View preparation and creation*
- *Application partitioning*

In other words, how do we handle a Web request, managing any necessary data access and generating presentation content for the user? How might we transform the incoming or outgoing stream? How do we determine what processing to perform to fulfill this request? What view should be used to service the request and how do we generate it? What are the logical abstractions and how do we design and decompose our software to take advantage of these abstractions?

Six presentation tier patterns are documented in the *Core J2EE Patterns Catalog*. The patterns and their tier

Every other month in this column we (Deepak Alur, John Crupi, Dan Malks, and other architects from the Sun Java Center (www.sun.com/service/sunps/jdc) will discuss various topics from our book, *Core J2EE Patterns: Best Practices and Strategies* (Alur, Crupi, Malks, Prentice Hall/Sun Press 2001). These topics include the 15 J2EE patterns in our catalog, design strategies, bad practices, refactorings, and pattern-driven design in J2EE technology.

categorizations are shown in Table 1. Because of the focus of this column, we’ll describe just one of the presentation patterns in detail. First I want to provide some context on how the presentation tier patterns fit within a common J2EE architecture, starting with a basic description of a system.

When a client makes a Web request for a particular resource, the request is processed, a view is generated, and a result is returned to the client. To provide more detail, we can define several logical components and subcomponents of a typical Web-based architecture:

- **Request handler:**
 - Pre- and postprocessing
 - Request processing
 - Command processing
- **Navigation and dispatch:**
 - Navigation resolution
 - Request dispatching
- **View processor:**
 - View preparation
 - View creation

These components and subcomponents are shown visually in Figure 1. Let’s have a look at the Intercepting Filter pattern, which addresses issues in the request-handling portion of the architecture, as outlined above and in the figure.

Intercepting Filter

The Intercepting Filter pattern documents issues relating to preprocessing and postprocessing a Web

request. Here are some common examples of preprocessing:

- **Decrypting an input stream:** The incoming data may have been encrypted for security purposes.
- **Decompressing a stream:** The incoming stream may have been compressed for more efficient transfer over the network.
- **Translating various encoding schemes to a common format:** Multiple encoding schemes require different handling. If each encoding is translated to a common format, the core request-handling mechanism can treat every request similarly. Some common examples of

TIER	PATTERN NAME
Presentation Tier	Intercepting Filter Front Controller View Helper Composite View Service to Worker Dispatcher View
Business Tier	Business Delegate Value Object Session Facade Composite Entity Value Object Assembler Value List Handler Service Locator
Integration Tier	Data Access Object Service Activator

TABLE 1 J2EE Patterns by tier

borland

www.borland.com

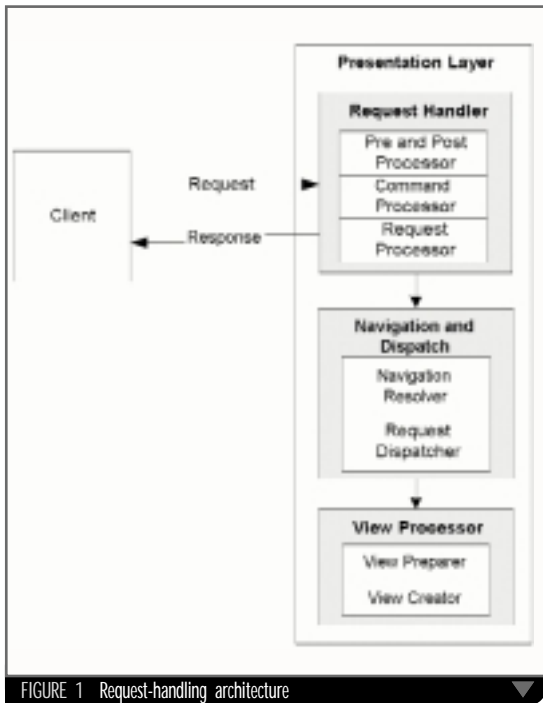


FIGURE 1 Request-handling architecture

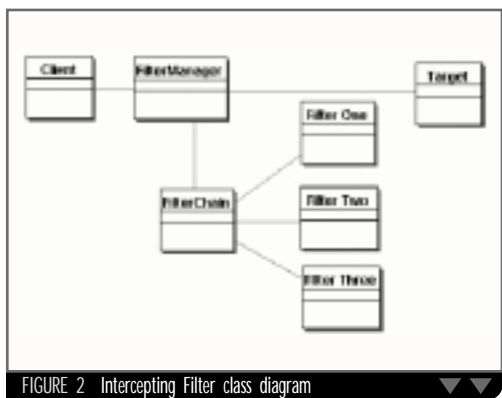


FIGURE 2 Intercepting Filter class diagram

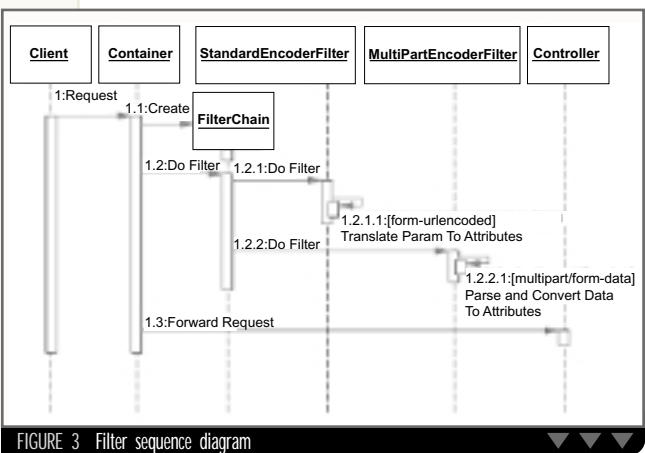


FIGURE 3 Filter sequence diagram

core processing flow, typically as part of the controller.

What constitutes postprocessing of a request? Here are some common examples:

- **Encrypting an output stream:** The outgoing data may be encrypted for security purposes.
- **Compressing a stream:** The outgoing stream may be compressed for more efficient transfer over the network.
- **Transformation of data for different clients:** Transformation into HTML, XML, or WML.

Additionally, one of the forces that motivates us to consider this

pattern is the desire to add and remove these processing components independently – independent of other filtering components and of the underlying core processing that fulfills the client’s request.

Let’s look at the class diagram (see Figure 2) that describes this pattern’s structure. In the figure the actual resource that is the target of the client request, such as a servlet or JavaServer Page technology, is represented by the Target class. The individual pre- or post-processing components that perform filtering functionality, as described above, are shown as Filter One, Filter Two, and Filter Three. One important thing to notice in this diagram is that there’s no direct association between any of the filters and the target resource. Additionally, there’s no direct association from any filter to one of the other filters.

There’s more than one way to approach any task. This is as true with software development as with anything else. So when I tell you that people approach the task of developing software in different ways, you certainly won’t be surprised. Some folks feel that most design work should precede implementation, while others like to jump in, write some code, and

// The fact is that wherever there is coding, there may be refactoring

There’s more than one way to approach any task. This is as true with software development as with anything else. So when I tell you that people approach the task of developing software in different ways, you certainly won’t be surprised. Some folks feel that most design work should precede implementation, while others like to jump in, write some code, and

providing pluggable behavior that can be used to decorate core request processing. Another benefit of using this pattern is that it promotes the reuse of these various filtering components across different requests, in different combinations, and even in different applications. There are several implementation strategies for this pattern, the most powerful of which leverages the standard filtering supported in the Servlet specification 2.3. Vendor support for this revision of the specification will be widespread in the not too distant future. Listings 1 and 2 are excerpts from the Standard Filter Strategy code example in the book. (The listings are available on the *JDJ* Web site,

www.sys-con.com/java/sourcec.cfm.) The example describes using filters to preprocess requests, checking their encoding schemes, and translating these different schemes to a common format. The common format is to store all request states within request attributes. Subsequently, any control code that checks for incoming values will get these values from request attributes, regardless of the original encoding. Figure 3 is the sequence diagram that shows the basic collaboration of the objects in the example. Note that in this implementation the role of the FilterManager from the class diagram is fulfilled by the Container in the sequence diagram. We hope this provides you with a basic understanding of the benefits and some implementing options for the Intercepting Filter pattern.

Refactoring

This is an important point, since it clarifies that the filters are loosely coupled both to the target resource and to other filters. This allows the filters to be easily added and removed unobtrusively, as mentioned.

Filters are an excellent way to layer functionality onto your system, pro-

There’s more than one way to approach any task. This is as true with software development as with anything else. So when I tell you that people approach the task of developing software in different ways, you certainly won’t be surprised. Some folks feel that most design work should precede implementation, while others like to jump in, write some code, and

- encoding schemes are:
- application/x-www-form-urlencoded
 - multipart/form-data
 - **Performing authentication or authorization:** Authentication and authorization may be performed either in a filter or as part of the

sprint soft

www.sprintsoft.com

PRESENTATION TIER REFACTORINGS

Introduce a controller
 Introduce synchronizer token
 Localize disparate logic
 Hide presentation tier-specific details from the business tier
 Remove conversions from view
 Hide resource from a client

BUSINESS AND INTEGRATION TIER REFACTORINGS

Wrap entities with session
 Introduce business delegate
 Merge session beans
 Eliminate inter-entity bean communication
 Move business logic to session

GENERAL REFACTORINGS

Separate data access code
 Refactor architecture by tiers
 Use a connection pool

TABLE 2 Refactorings

start to think about how these bits of implementation fit together. The difference is basically that of top-down design versus bottom-up design.

Refactoring applies to either approach, though it's typically applied in an environment where there is an understanding that design is spread across the life of the project. That said, the fact is that wherever there is coding, there may be refactoring. Martin Fowler, in his great book *Refactoring: Improving the Design of Existing Code* (Addison-Wesley), describes refactoring as "improving the design of the code after it has been written." His book identifies many common design flaws and describes the incremental coding changes that result in improved design. The issues are typically general and not specific to any particular area of Java or software development.

The lion's share of Fowler's book is devoted to what he calls "small refactorings," meaning the design changes are at a very low level, each involving several discrete coding modifications, such as adding a parameter to a method. A small portion of the book, coauthored by Kent Beck, is devoted to "big refactorings," which exist at a higher level of abstraction and have steps that aren't as well defined or as concrete.

In our book we include some J2EE technology-specific refactorings, describing opportunities to improve the design

of a J2EE technology-based system and the relevant steps involved. The format and style is based on that in Fowler's book, which we find extremely valuable. Based on Fowler and Beck's definition, the J2EE refactorings included in our book might be called "medium refactorings" based on their level of abstraction. The refactorings are listed in Table 2, categorized by tier.

We find these refactorings to be excellent companions to the patterns and bad practices described in the rest of our book. In fact, you can think about the refactorings as often providing the steps that help guide the developer from a less optimal solution, or bad practice, to a more optimal one, suggested by a pattern.

In a future article we'll provide more information on these refactorings and their relationship to the patterns in the catalog. We'll also go into greater detail on the presentation, business, and integration tiers, as well as communication across these tiers.

Thank you for reading, and please e-mail us at CoreJ2EEPatterns@sun.com to provide feedback on this article and to suggest other topics of interest. ☛

AUTHOR BIO

Dan Malks, a senior Java architect with Sun Microsystems, is currently focusing on distributed, service-based designs, patterns, and implementations. He has developed in a variety of environments, including Smalltalk and Java, while focusing on OO technologies. Dan has published articles on Java in leading industry periodicals, and holds bachelor's and master's degrees in computer science.

Portions of this article contain excerpts with permission from *Core J2EE Patterns* by Deepak Alur, John Crupi, and Dan Malks (ISBN 0-13-064884-1) Copyright 2001. Sun Microsystems, Inc.

Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Sun, Sun Microsystems, the Sun logo, Java, J2EE, Java Center, and JavaServer Pages are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. Sun Microsystems, Inc., may have intellectual property rights relating to implementations of the technology described in this article, and no license of any kind is given here. Please visit www.sun.com/software/communitysource/ for licensing information.

The information in this article (the "information") is provided "as is," for discussion purposes only. All express or implied conditions, representations, and warranties, including any implied warranty of merchantability, fitness for a particular purpose, or non-infringement, are disclaimed, except to the extent that such disclaimers are held to be legally invalid. Neither Sun nor the authors make any representations, warranties, or guarantees as to the quality, suitability, truth, accuracy, or completeness of the information. Neither Sun nor the authors shall be liable for any damages suffered as a result of using, modifying, contributing, copying, or distributing the information.

dan.malks@sun.com

SYS-CON MEDIA

PUBLISHER, PRESIDENT, AND CEO
 FUAT A. KIRCAALI fuat@sys-con.com
 VICE PRESIDENT, BUSINESS DEVELOPMENT
 GRISHA DAVIDA grisha@sys-con.com

ADVERTISING

SENIOR VICE PRESIDENT, SALES AND MARKETING
 CARMEN GONZALEZ carmen@sys-con.com
 VICE PRESIDENT, SALES AND MARKETING
 MILES SILVERMAN miles@sys-con.com
 ADVERTISING SALES DIRECTOR
 ROBYN FORMA robyn@sys-con.com
 ADVERTISING ACCOUNT MANAGER
 MEGAN RING megan@sys-con.com
 ASSOCIATE SALES MANAGERS
 CARRIE GEBERT carrieg@sys-con.com
 KRISTIN KUHNLE kristen@sys-con.com
 ALISA CATALANO alisa@sys-con.com

EDITORIAL

EXECUTIVE EDITOR
 M'LOU PINKHAM mpinkham@sys-con.com
 EDITOR
 NANCY VALENTINE nancy@sys-con.com
 MANAGING EDITOR
 CHERYL VAN SISE cheryl@sys-con.com
 ASSOCIATE EDITORS
 JAMIE MATUSOW jamie@sys-con.com
 GAIL SCHULTZ gail@sys-con.com
 ONLINE EDITOR
 LIN GOETZ lin@sys-con.com

PRODUCTION

VICE PRESIDENT, PRODUCTION AND DESIGN
 JIM MORGAN jim@sys-con.com
 ART DIRECTOR
 ALEX BOTERO alex@sys-con.com
 ASSOCIATE ART DIRECTORS
 LOUIS F. CUFFARI louis@sys-con.com
 CATHRYN BURAK cathyb@sys-con.com
 RICHARD SILVERBERG richards@sys-con.com
 AARATHI VENKATARAMAN aarathi@sys-con.com

WEB SERVICES

WEBMASTER
 ROBERT DIAMOND robert@sys-con.com
 WEB DESIGNERS
 STEPHEN KILMURRAY stephen@sys-con.com
 CHRISTOPHER CROCE chris@sys-con.com

ACCOUNTING

CHIEF FINANCIAL OFFICER
 BRUCE KANNNER bruce@sys-con.com
 ASSISTANT CONTROLLER
 JUDITH CALINAN judith@sys-con.com
 ACCOUNTS RECEIVABLE
 JAN BRAIDECH jan@sys-con.com
 ACCOUNTS PAYABLE
 JOAN LAROSE joan@sys-con.com
 ACCOUNTING CLERK
 BETTY WHITE betty@sys-con.com

SYS-CON EVENTS

VICE PRESIDENT, SYS-CON EVENTS
 CATHY WALTERS cathyw@sys-con.com
 CONFERENCE MANAGER
 MICHAEL LYNCH mike@sys-con.com
 SALES EXECUTIVES, EXHIBITS
 MICHAEL PESICK michael@sys-con.com
 RICHARD ANDERSON richard@sys-con.com

SHOW ASSISTANT

NIKI PANAGOPOULOS niki@sys-con.com
 REGISTRATION ASSISTANT
 JACLYN REDMOND jaclyn@sys-con.com

CUSTOMER RELATIONS / JDJ STORE

MANAGER, CUSTOMER RELATIONS / JDJ STORE
 ANTHONY D. SPITZER tony@sys-con.com
 CUSTOMER SERVICE LIAISON
 PATTI DELVECCHIO patti@sys-con.com

iona

www.iona.com

Getting Focus()ed — and a Quick JavaScript Lesson



WRITTEN BY
CHARLES
AREHART

Many new J2EE developers get caught up in focusing on the details and nuances of servlets and JSPs and, as a result, may not learn how to leverage JavaScript. Some may even dismiss it as too much hassle, given cross-browser compatibility issues.

For both audiences there's still value in learning at least a minimal amount about client-side scripting. Even learning about just one feature – setting the cursor on the first form field you may have – can bring measurable benefit to your site visitors.

In this month's **Journeyman J2EE**, we depart from pure J2EE topics to address this subject, which should be understood by any Web application developer. If you're a newcomer to JavaScript, I'll provide more than enough to get started.

Where's Your Focus?

Here's the motivation for the problem I want to solve: you visit a Web site that offers a form inviting you to enter some input. No big deal – we see them all the time, right? Now, how do you get to the point of entering data into that first field?

Do you use your mouse to move the cursor to that first data entry area? Or are you a keyboard maven, like me, in which case you prefer to tab to such a field? The problem is you may have to hit the tab key several times before you can enter data.

Indeed, it's typical to visit sites with navigational toolbars across the top and/or left side of the form, forcing you to tab dozens and dozens of times. It's an annoyance, especially if visitors to your site know the site designer could have easily prevented the problem.

If you think that's being pedantic, consider a simpler example. If you show users a page that clearly is expecting them to enter some data (perhaps a user ID or a search argument, etc.) before proceeding, why not put the cursor right on the data entry field rather than forcing them to use the mouse or keyboard to get there. It's a usability issue.

There's an incredibly simple solution. Just the tiniest bit of JavaScript is

needed, a single line of code, really. This article shows how to use it and also lays the most basic foundation for using JavaScript, if you're new to it.

Laying the Foundation

The technique we're talking about involves putting the "keyboard focus" on whatever form input field – a text, password, or textarea box or even a radio, checkbox, or select field – you want the user to enter data into first. The solution is the JavaScript "focus" method.

The JavaScript language is available to any Web page designer on nearly all Web browsers. (Microsoft calls its version of the language JScript. They're nearly the same, especially for the purposes of this article.) It's a scripting language that can be used to add features to your Web page that aren't otherwise provided by HTML.

This article shows you the simplest application of this focus method. It's pretty straightforward and should work fine in most instances. JavaScript is widely supported by most browsers now, and this particular feature is supported identically in both Netscape's and Microsoft's versions of the language, so there's no need to worry about it breaking on different browsers.

Gimme the Goods

We need to put this method in some JavaScript code inside our page. If you haven't coded JavaScript on your pages, there are several ways to do this. You can embed the code in a pair of <SCRIPT> tags (and even then you have choices about where those tags are placed) or you can place it in special attributes of HTML tags (such as ONLOAD within a <BODY> tag).

Let's take a simple example in the first of those two forms. We can use the JavaScript alert() method to cause a message to be displayed to the user in a popup box. To do that, we could use the following code:

```
<SCRIPT LANGUAGE="JavaScript"
TYPE="text/javascript">
alert('Hello World');
</SCRIPT>
```

Put that code almost anywhere in an HTML page and when you view that page in a browser, you'll see a message pop up: "Hello World." Try it. We can use the same approach to specify JavaScript code to indicate that we want to place the focus on a particular field. We're almost there.

Focus on Forms

The focus() method operates on form fields. One of the many powers of JavaScript is that not only can we refer to (display, test) all the elements of our page (all the forms, form fields, indeed every tag and its contents and attributes), we can manipulate them as well.

How do we specify that we intend to work with some part of our page, such as a particular form field? Most of you know that you can give a name to a form field, and in JSP/servlets that will become the name of an attribute in the request scope on the form's action page. In our JavaScript we can use that same name to refer to the form field to give it focus. If we have an input field for a "UserId" that should be entered, the HTML would be:

```
<FORM ->
-
Enter UserId: <INPUT TYPE="text"
Name="UserId">
-
</FORM>
```

Of course, the <FORM> tag would be specified completely and other form field information would be provided both before and after this input field. However, it's important to know that this does indeed (and must) occur within a <FORM> tag.

A win-win situation



J2ME



J2SE



J2EE



Home

altoweb

www.altoweb.com



J2ME



J2SE



J2EE



Home

ibm new

One reason it's important (besides being the way HTML is coded!) is that when we want to refer to a particular element within a Web page from within JavaScript, we need to refer to it in the form of its relationship to the entire page. This input form field, "UserId", occurs inside of a form. On a simple level we might refer to it as form.UserId.

This isn't technically correct, however, since it's possible to have more than one form in a page. Indeed, there are rules that describe how to refer to elements on a page within JavaScript. It's called the Document Object Model (DOM). In any case, we need to indicate the specific form in which the field occurs.

What's in a Name?

There are two ways to do this: we can refer to the form by name or by indicating the relative location of the form on the page. Which is appropriate for you depends on your situation. If you've specified a NAME attribute on the <FORM> tag, you'd use that name.

If we had specified NAME="Login" on the <FORM> tag, we would refer to the form field as Login.UserId. But we're not done yet. The form itself occurs within the Web page, and while it might not seem necessary, the contents of the page are considered to be within the "document" on the page (again, recall the "Document" Object Model). We finally have the complete means by which to refer to the field: document.Login.UserId.

Before leaving this subject, let's clarify that if you haven't named your form, you can still refer to it by indicating the relative location of the form within the document. There's a special "forms" array in every Web page document, so if you have only one form in yours, you would indicate it as the first form.

Unfortunately, even this can trip you up because in JavaScript (as in Java) you start counting lists of elements at 0 rather than 1. So the way to refer to your form field (assuming it's within the first or only form on the page) would be:

```
document.forms[0].UserId
```

Notice that this isn't used as "form[0]" but as "forms[0]". It's a subtle point, but if it's not specified correctly, you'll receive an error.

We're on the Case

Indeed, it's also critically important to remember that JavaScript is case-sensitive. The case you choose when naming something must be the case you use when referring to it later with JavaScript.

If we were to refer to our form field as Document.Forms[0].UserId, for example, we'd get an error. JavaScript expects us to refer to both the "document" and "forms" elements as all lowercase.

Similarly, since we named the form field UserId, we must refer to it exactly that way. If we referred to it as document.forms[0].userid, or even document.forms[0].UserId, we'd receive an error either way.

Forcing the Focus()

To set the focus we use the focus method just like a reference to an object's method in Java. To set the focus for our form field called "UserId", we might specify it as:

```
document.forms[0].UserId.focus()
```

That's about it. There's nothing to be specified within the parentheses. (Notice that the word focus is also all lowercase.) This statement, when executed, will cause the named field in the named form to receive the focus when the page is displayed to the user. In other words, the cursor will be resting on that field when the page is loaded.

Our final challenge is to decide where and how to specify this statement. Of course, it's JavaScript and must be specified to be executed as such. We mentioned earlier that this could be done by way of the <SCRIPT> tag. The following code will cause the focus to be placed on our UserId field:

```
<SCRIPT LANGUAGE="JavaScript"
TYPE="text/javascript">
<!--
document.forms[0].UserId.focus()
//-->
</SCRIPT>
```

If only it were this easy. Well, it is – almost! You just need to make sure the statement is executed after the form has been loaded. In JavaScript it's inappropriate to refer to a variable (or form field, as it were) before it's been defined. In this case, with this approach to executing the statement, we need to be sure not to put this code on our page prior to the form itself.

There are other ways to specify JavaScript within a page. Besides the <SCRIPT> tag, we can also use the event-handler attributes of other tags. For instance, we could also force the JavaScript to execute only after the entire page has loaded. Some may know that there's an attribute for the BODY tag called ONLOAD that does just what we need, causing whatever statements it executes to be run only after the page has loaded. Since the form is within the page, this is indeed another solution to our problem.

Fortunately, we can easily specify our JavaScript statement (the statement itself, not the <SCRIPT> tags) by placing it in the BODY tag's ONLOAD attribute, as in:

```
<BODY
ONLOAD="javascript:document.forms[0].
UserId.focus()">
```


ibm

www.ibm.com



J2ME



J2SE



J2EE



Home

Pretty nifty! (The "javascript:" directive preceding the statement is not usually required, but it's the formal method of specifying JavaScript within an attribute like ONLOAD.)

Experienced developers may prefer yet another approach: creating a function instead and calling that function in the ONLOAD.

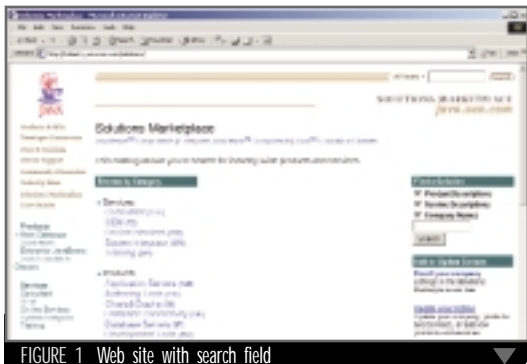


FIGURE 1 Web site with search field

But Will It Fail?

You should have no trouble if you:

1. Specify the JavaScript statement using one of the two appropriate approaches.
2. Refer to the form field using the naming references indicated.
3. Remember to specify the proper case for the naming reference.
4. Ensure that the statement occurs only after the form has been loaded.

There are a couple of other edge cases to be aware of, however. First, some folks use a single page to serve as both the form and action page (loaded from within the same JSP template, for instance). That's fine and can be a powerful way to reuse lots of code, but be careful: if you leave JavaScript code referring to the form field on the page, but the output of the action page (POST) processing doesn't show the form (because you're showing the results of processing the form, not the form), you'll get an error. This is particularly risky when using the ONLOAD approach, because you may not think of it as serving both the form and action page when coding the page.

In a similar vein, and equally tricky in some situations, you don't want to include the JavaScript statement on the page if the form field to which it's referring isn't included in the page. With dynamically built pages as in JSP and servlets, this isn't as strange as it might seem. Forewarned is forearmed.

What if a user's browser simply doesn't support JavaScript? There's very good news.

With the <SCRIPT> tag approach described earlier, you may notice that just inside the opening and closing

<SCRIPT> tags are HTML comment tags. This is a standard JavaScript approach that guarantees that if the browser doesn't support the <SCRIPT> tag (which it will simply ignore), it will also ignore the JavaScript statements inside the script and comment tags. However, browsers that support JavaScript still execute the statements inside the comment tags.

With the ONLOAD approach as well, browsers that ignore JavaScript also won't recognize JavaScript-related attributes. The ONLOAD attribute is purely and simply for executing JavaScript and hence is ignored by non-JS browsers. By association, the value specified for that attribute (our JavaScript statement in the second approach) is also ignored.

Some Caveats

Should you go ahead and add the focus method for the first input field of the forms in all your pages? Well, first it may not be appropriate to the interface. On some pages the form and its input fields may not really be the focus of the page. Consider a page that happens to have a search field on it, such as <http://industry.java.sun.com/solutions/> (see Figure 1).

If you look closely at the top right corner, there is indeed a "form" on this page for searching the site, but it's definitely not the focus of the page. There's also a search field to "Find a Solution" on the right-hand side. Should we put the focus in the first form field? The second? Either? Maybe neither. If you put it on the second, which may seem a focus of the page, keyboard users would find that tabbing once wouldn't take them to the top nav bar but instead to the links under that search field. That's not good usability, either.

Another reason not to blindly set the focus on any first form on the page is if the form doesn't appear at the top of the page. If we did that, an unintended result would be that the screen would be scrolled down when displayed, so that the field with focus (that form input field) was visible. This may cause the top of the page to be hidden from the user. Remember, too, that users often have different (and lower resolution) monitors than developers, increasing the chances of this problem developing. It can also happen if the users don't maximize their screen when displaying your page.

When would it be appropriate to put the focus on a field? When the form is clearly the focus of the page. Going back to the previous example at Sun's Solutions Marketplace, consider the link under that "Find a Solution" button, labeled "Enroll Your Company." On the page that will be shown, it prompts for Company Name.

That's clearly the focus of the page, so putting the focus on that field would make sense (sadly, they're not setting the focus using this approach as of this writing).

If you visit www.altavista.com, a popular search engine, you'll see that searching is clearly the focus of the page, and they do indeed use the focus method so that you can start typing search criteria as soon as the page appears.

Another natural example, and one that nearly all developers end up dealing with, is a login page that's presented to enter a site or access secured areas. If the user can't proceed without entering a user ID/password and the prompt for this is the only reason the page is being presented, there's little reason not to set the focus on the UserId field.

Afterword: Using SCRIPTJ in JRun Studio

There's one last useful trick that will be of interest to users of JRun Studio (a similar feature may exist in other IDEs). It also works in its sister Macromedia products, ColdFusion Studio and HomeSite.

Remember the <SCRIPT> tags offered earlier with the HTML comment characters specified within them – I mentioned that these are a standard set of tags to be used for all JavaScript entered within a page (other than that entered within other tag attributes like ONLOAD).

You could hope to remember to enter that properly formatted set of basic tags:

```
<SCRIPT LANGUAGE="JavaScript"
TYPE="text/javascript">
<!--

//-->
```

or you could take advantage of a nifty shortcut in Studio: simply type SCRIPTJ, press CTRL-J (the control key and J), and watch as Studio converts that into the set of tags offered above. The cursor is even sitting there within the paired tags waiting for you to type in your JavaScript. (Hey, now there's a great use of cursor focus!)

Summary

Using the focus method is a win-win that should be used in many Web forms. JavaScript is ignored in browsers that don't support it or have it turned off, and being a JavaScript 1.0 element it works even in Netscape 2 and IE 3. Plus, it really is just a single statement of JavaScript code in nearly all cases. Now do you see why I find it so frustrating when a site doesn't use it? I hope newcomers to JavaScript have also learned a few things. ☺

carehart@systemanage.com

AUTHOR BIO
Charles Arehart is a 20-year IT veteran with experience spanning a range of technologies, including large scale database systems. For the past four years he's been an active trainer, writer, and consultant in enterprise Web application development. He contributes to several resources, provides on-site coaching and consultation, and is a frequent speaker at user groups throughout the country.

bea
www.bea.com



KEITH BROWN J2SE EDITOR

Java Beyond the Server: It's Time to Really Swing

Happy New Year! I trust you had a good festive break...not drinking or eating too much. Who am I? Good question. My name is Keith Brown and I'm the new J2SE editor of *Java Developer's Journal*. As this is the first issue of the new year, our editor-in-chief felt it was the perfect time for me to kick off and bring my flavor of Java to you each month.

My professional history is varied, so the least said about that the better (only joking). I'm a seasoned developer with nearly ten years of software engineering behind me, five of them spent working with this beautiful language we call Java. I've been involved with Java at many levels, mainly at the client-side, as opposed to the trendier jet set who specialize at the server-side. When Alan first spoke to me regarding this role, I was excited about his vision for the magazine. Since it aligned with my own beliefs regarding software engineering, I felt I could take this role and do it justice.

As the newest member of the team, I would like to spend a little time discussing my own personal views on the state of the industry, and where we can work together to make sure we are suitably armed with the tools to implement solutions for the next wave of problems that are winging their way toward us.

Did you catch last month's **Guest Editorial** from Eric Shapiro? Eric managed to encapsulate the very mood of the J2SE movement. As he said, people are quick to automatically dismiss Java to the server-side, automatically reaching for infamous phrases such as EJB, JSP, and JMS. But Java is more than just server-side – look at our J2ME section if you have any doubts.

I'm here to help reinforce the case for Java outside of the server world. Swing has had a bad rep to deal with; in fact, the history of GUIs in Java isn't a pretty one. When Java first burst onto the scene we were stuck with AWT. When Windows and even X-Windows had all these fantastic APIs, Java developers were definitely at the bottom of the heap with AWT. But it has to be said that the efforts of developers to produce beautiful client-side applications were admirable considering the tools they had to work with.

There was hope. We were told of a GUI API that would allow us to get to the tools we needed to produce production-quality, client-side applications, a GUI that would offer the flexibility and speed of any native platform-specific tool, a GUI to finally bring Java to the masses. Wow – sounded too good to be true.

Sadly it was. What finally arrived was something that was definitely a far cry from the pipe dream we were all living in. To say Swing was a disappointment to some would be a bit of an understatement. The system was buggy, slow, and very inflexible to porting. Maybe it was rushed out the door too quickly and people expected too much from it initially. After all, the APIs it was competing against had been seasoned over many years with a huge amount of testing and development invested in them. Swing was finding its feet with a very demanding audience.

I think that's one of the reasons why Java's popularity grew so quickly at the server-side. Technologies such as servlets and then JSPs made it very easy to build front ends to applications that could be quickly and easily deployed to the masses

Java Beyond the Server

Swing has never managed to quite get there, but after a number of years in development, the time has come to start delivering on some of its promises.

by Keith Brown

44

Using Regular Expressions in J2SE 1.4

Even a regex neophyte should have no trouble with the pattern and the matcher.

by David Weller

46

Using Assertions in Java

Building confidence in your code

by Jonathan Amsterdam

52

Synchronizing Java Threads on a Shared Resource with Multiple Views

A simple and elegant solution

by Vishal Goenka

60

Evolutive Java Applications

Enable functionality-driven software architecture

by Alvaro Schwarzbarg

66

without worrying about extra downloads – in other words, the very premise that Swing was hoping to solve, but never quite got there due to its size. Even today, if you surf to a page that has a Swing applet, the browser goes into a bit of a fit; if you're really unlucky, you'll have just incurred a 10MB download, worse case.

Swing has never managed to quite get there, but after a number of years in development, the time has come to start delivering on some of its promises. ☻

keith.brown@sys-con.com

AUTHOR BIO

Keith Brown has been involved with Java for many years. When he's not coding up client solutions for a European Java company, he can be found lurking in the corridors of conferences all around the world.

dice
www.dice.com

Using Regular Expressions in J2SE 1.4

Even a regex neophyte should have
no trouble with the pattern
and the matcher



WRITTEN BY
DAVID WELLER

A regular expression (regex) is an essential part of software development. Indeed, the programming language Perl is, in effect, a language written around a regex parser.

This article focuses on Sun's implementation of a regular expression package, `java.util.regex`. In addition, this article also assumes you have some familiarity with regular expressions (if not, see the sidebar for a brief introduction). Each regular expression used here, however, is fully described, so even a regex neophyte should have no trouble.

The regex package is divided into two major packages: `pattern` and `matcher`. Before going into the details of these packages, it's important to understand their relationship.

A *pattern* contains a specific regular expression that's created by compiling a regex string. If the string doesn't compile, the pattern isn't valid and a `PatternSyntaxException` is thrown. You should always have a `try/catch` block when compiling regexes to catch this exception. For reasons of brevity, exception handling is excluded from the examples here.

A *matcher* is an object that does the real "grunt work." It holds a reference to the input stream (of type `CharSequence`, like `String` or `StringBuffer`) and keeps all sorts of stateful information about the results of a pattern search, string start/end locations, etc. A `matcher` object can only be created through an instance of a `pattern`.

The most basic creation sequence is:

- **Compile a regex:** If it's valid, it will return a `pattern`, otherwise it will throw a `PatternSyntaxException`.
- **Create a matcher:** Give the `pattern` instance an input set to match against.

Once you have a `matcher`, there are three Boolean functions you can use to determine if the input conforms to the `pattern`. Of the three you will typically

use, the `find()` method returns a Boolean the moment it matches the regex. The `matcher` will then also remember where the last match was and will pick up from that point with the next `find()` method (for those of you familiar with Perl, this behavior is similar to the `"g"` modifier at the end of your regex).

Our first example performs the basic creation sequence mentioned above, then calls `find()` repeatedly to count

the number of times the word "fish" exists in the input sequence (see Listing 1).

While the example itself isn't interesting, the above "code pattern" occurs frequently when writing software that uses the regex packages. Make a point to remember it!

Now we'll see how we can replace a matched pattern with something else. There are two mechanisms to change an input string: `appendReplacement()` and

WHAT IS A REGULAR EXPRESSION?

These next few paragraphs will not come close to doing justice to the full scope and power of regular expressions, so let me first recommend Jeffrey Friedl's amazing *Mastering Regular Expressions* (O'Reilly). This book contains everything you need to know about the mysterious regex.

Many developers, particularly in the UNIX world, are familiar with the tool "grep" (which stands for Global Regular Expression Print). I've found, however, that many grep users are also unfamiliar with what real regular expressions are.

Simply put, regular expressions are a way to define a pattern that can match a sequence of characters. The most basic regex is a literal string, like "cat". It will match every time it sees the sequence "cat" in the input. Moving up in complexity, the period (.) will match any single character, so the regex "c.t" will match sequences, such as "cat", "cot", "c2t", and even "c t". If we wanted to limit the choices, we could use a selection range instead, which is a sequence of characters enclosed in brackets. So the regex, "c[au]t" would match "cat", "cot", or "cut".

You can also find special characters using "metacharacters". For instance, the metacharacters `\n` and `\t` will match a newline and tab, respectively. This brings us to an interesting point: if the backslash is used to signal a metacharacter, how do you search for a backslash? Simple. To search the input for a character that would be considered a special character, simply "escape" the character with a backslash. The regex `\\.` would search the input for the character sequence `\"`.

Every character or character group (part of a regex surrounded by parentheses) can be modified by "quantifiers". The quantifier `"*"` means "zero or more times", and the quantifier `"?"` means "once or not at all". So the regex `"c([aou]*)t"` will match "cat", "coot", "cououaaoot", etc. The regex `"coo?t"` will match "cot" or "coot" only.

Let's wrap up by listing a short table of special metacharacters that are frequently seen in regular expressions:

| | |
|-----------------|--|
| <code>^</code> | beginning of a line |
| <code>\$</code> | end of a line |
| <code>\d</code> | a digit |
| <code>\D</code> | a nondigit |
| <code>\s</code> | a whitespace character (space, tab, newline, etc.) |
| <code>\S</code> | a nonwhitespace character. |

altova

www.altova.com

Group 1 Group 2 Backreference to Group 1

“<(.*)>(.*?)</\1>”

FIGURE 1 Regex string ▼

appendTail(). Understanding how these two work together is tricky, but mastering this relationship is critical. Let's look at how the two work together to perform a simple replacement.

Let's use the pattern “fish” and the input “The fish in the hat”. We want to replace the pattern with “cat”. We first create a pattern and matcher, then do a find. When the find returns true, we create a new StringBuffer to hold our modified string (never modify the matcher's input string directly!), then begin changing our output string.

At this point, it's important to remember that our Matcher instance, “m”, now knows the start and end point of the most recent match. It also knows the end point of the previous match (which is “0” if there is no previous match). When appendReplacement is called, it uses this information to perform the following two steps:

1. It appends everything from the end of the previous match-up to the beginning of the first match (in our example, that's simply “The”) to the output string.
2. It then appends the replacement string onto the output string. Our output string is now “The cat”.

Finally, we call appendTail, which replaces the remainder of the original input string (“in the hat”) to the output string, yielding the expected result (see Listing 2).

The preceding example is only useful for single replacements, which isn't very realistic. Let's modify this example to replace all occurrences of a pattern by using a while loop (see Listing 3).

Conveniently, the matcher class has a “replaceAll” method that will do exactly what the preceding code will do. It is, however, only useful for simple string replacements.

Now that you know the basics of searching and replacing, let's look at the more powerful features of the regex package – using groups and quantifiers.

The simple pattern, “one.*two”, is typically read (comprehended) as: “The sequence ‘one’, followed by any number of any kind of character, followed by

“two”. Because we have used the “greedy” quantifier, “*”, this regex will get the largest match it can find. Be careful using quantifiers, as they can yield strange results. Mastery of quantifiers, however, is essential to writing good regular expressions. Jeffrey Friedl's book, *Mastering Regular Expressions*, (O'Reilly) contains numerous examples of the many different forms of quantifiers, and I recommend reading it to learn more.

Back to our example, calling find would result in the matcher “marking” these places (noted by the arrows): “_one if by Java, two_ if by C”. This is interesting if we want to replace the entire regex, but what if we want to replace only the characters between the “one” and the “two”? This is where grouping comes in.

Grouping is the most powerful feature of the regex package because it allows us to manipulate sets of substrings. If we change the regex to include the grouping markers (open/closed parentheses), we then create nested groups. These groups start with the number 1; group(0) is always the entire matched pattern. If we used the regex “one(.*)two”, we would generate a “group list” in our match.

Let's look at a simple example now, as seen in Listing 4.

Note that the for loop uses “<=”, rather than the traditional “<”.

Our results are:

```
>java ShowGroups
Group(0) is "one if by Java, two"
Group(1) is " if by Java, "
```

Let's wrap this up by looking at how you would write a simple XML-style tag parser.

First, study the regex string in Figure 1. This regex introduces a “backreference”, which is when a group is self-referenced in a regex. In this example, the first group (which matches a tag name) is later used as a backreference, expressed as a backslash followed by a group number. This is a powerful and useful feature of regular expressions.

The first group is the first “(.*?)” that

you see. It is used to initially “guess” that it sees a tag. The regex, however, says it's not a tag unless that same string is on the tail end, which is where you see the backreference “\1”.

At this point, you should notice a strange quirk of Java. In normal regex strings, the backreference would just look like “\1”, but since escaped characters are interpreted by the Java compiler (e.g., “\n”), we must use a “double backslash” to signify a regex “escape”. You must test your Java regex strings carefully, as this single quirk can cause you hours of grief (to which this author can testify!).

Now let's look at the code. It's really a simple example of recursion. The findTag method is simply handed an initial input string. When the pattern matches, everything inside the tag boundaries (Group 2) is again handed to findTag, and the parsing starts again (see Listing 5).

When we run the program, we get:

```
>java tagParser
Found tag: bold, inner string =
<italic>bold-italic</italic>
Found tag: italic, inner string =
bold-italic
```

I encourage you to look at other regular expression packages if you intend to do extremely complex regex work. The most notable example is the ORO package freely available through the Apache Jakarta Project (<http://jakarta.apache.org/oro>). It's more full-featured than the JDK regex package, but the usage is similar. Sadly, Sun decided not to implement the regex package through interfaces, making it (currently) impossible to freely switch between the JDK and ORO regex packages. On the brighter side, the Sun regex package is full-featured enough for places where a typical regex engine is needed.

In conclusion, I hope I've helped you to understand how to use the power of the regex classes. The graceful combination of the pattern and matcher classes helps maintain a separation of concerns. This addition to Java has been long overdue. Have fun, and happy parsing! 🍀

Acknowledgment

The author would like to acknowledge the gracious feedback of Roger Moore of Valtech Technologies (Dallas) and Tom Wood of Valtech Technologies (Houston).

▼ ▼ david.weller@valtech.com

AUTHOR BIO

David Weller is a principal managing consultant at Valtech Technologies, Inc., an international consulting firm specializing in .NET/J2EE/Unified Process development, skills transfer, and training. He holds a computer science degree from the University of Houston at Clear Lake.

insession

www.insession.com

Listing 1

```
import java.io.*;
import java.util.*;
import java.util.regex.*;
public final class Simple {
    public static final void main(String args[]) {
        Pattern p = Pattern.compile("fish");
        Matcher m = p.matcher("one fish, two fish, red fish,
        blue fish");
        int count = 0;
        while (m.find()) {
            count++;
        }
        System.out.println("Count = " + count);
    }
}
```

Running this program yields:
D:\projects\regex>java simple
Count = 4

Listing 2

D:\projects\regex>java Replace
The cat in the hat

Here's the code itself:

```
import java.io.*;
import java.util.*;
import java.util.regex.*;
public final class Replace {
    public static final void main(String args[]) {
        Pattern p = Pattern.compile("fish");
        Matcher m = p.matcher("The fish in the hat");
        StringBuffer output = new StringBuffer();
        if (m.find()) {
            m.appendReplacement(output, "cat");
            m.appendTail(output);
        }
        System.out.println(output.toString());
    }
}
```

Listing 3

```
import java.io.*;
import java.util.*;
import java.util.regex.*;
public final class ReplaceAll {
    public static final void main(String args[]) {
        Pattern p = Pattern.compile("fish");
        Matcher m = p.matcher("One fish, two fish, red fish,
        blue fish");
```

```
StringBuffer output = new StringBuffer();
while (m.find()) {
    m.appendReplacement(output, "cat");
}
m.appendTail(output);
System.out.println(output.toString());
}
}
```

Listing 4

```
import java.io.*;
import java.util.*;
import java.util.regex.*;
public final class ShowGroups {
    public static final void main(String args[]) {
        StringBuffer input = new StringBuffer("one if by Java,
        two if by C");
        Pattern p = Pattern.compile("one(.*)two");
        Matcher m = p.matcher(input);
        while (m.find()) {
            for(int i=0; i <= m.groupCount(); i++){
                System.out.println("Group("+i+") is \"" +
                m.group(i) + "\"");
            }
        }
    }
}
```

Listing 5

```
import java.io.*;
import java.util.*;
import java.util.regex.*;
public final class tagParser {
    public static final void main(String args[]) {
        Pattern p = Pattern.compile("<(.*?)>(.*?)</\\1>");
        String input = "This is <bold><italic>bold-italic</
        italic></bold>";
        findTag(p, input);
    }

    private static final void findTag(Pattern p, String in) {
        Matcher m = p.matcher(in);
        boolean result = m.find();
        while (result) {
            System.out.println("Found tag: " + m.group(1) + ",
            inner string = " + m.group(2) );
            findTag(p, m.group(2));
            result = m.find();
        }
    }
}
```

Download the Code!
www.javadevelopersjournal.com

int

www.int.com

loox

www.loox.com

Using Assertions in Java

10

Building confidence in your code

01111001001

Written by Jonathan Amsterdam

Java's new assertion mechanism, a welcome addition to the language now available in version 1.4, allows programmers to increase the robustness of their code by sprinkling it liberally with assert statements. The new assertion feature is easy to use, but any language feature, no matter how simple, can be used well or poorly. Here I'll explain how to use Java's assertion facility, and how not to misuse it.

Using assertions couldn't be easier. Anywhere you can put a statement in Java, you can now write

```
assert boolExpr;
```

where `boolExpr` is any Boolean expression. If the expression is true, execution continues as if nothing happened. If it's false, an exception is thrown. You can disable assertions at runtime if you wish, effectively removing them from your code.

Why Use Assertions?

Assertions are a cheap and easy way of building confidence in your code. When you write an assertion, you're enabling the machine to check your beliefs about your program. You write the assertion thinking that it's true, but as you well know, not all of your beliefs about your code are true - if they were, you wouldn't have any bugs. Assertions can help uncover bugs early on.

For example, here's a bit of code that makes a choice based on the remainder of dividing n by 3:

```
if (n % 3 == 0) {  
    ...  
} else if (n % 3 == 1) {  
    ...  
} else {  
    assert n % 3 == 2;  
    ...  
}
```

Now obviously, $n \% 3 == 2$ in the final else of this statement, since the remainder when you divide a number by 3 is either 0, 1, or 2. So there's no need to do an explicit test. But these new assert statements are easy to write and you can always disable them, so you add one just for kicks. That will turn out to have been a wise choice when the code is run with a negative value of n . The Java % operator, like the mod operator of most programming languages, gives negative results when its left operand is negative: $-5 \% 3$ is -2, not 2. The assertion will fail, the program will stop, and you can easily correct both the code and your false belief about how % works.

Here's another example from some recent code of my own:

```
TransactionEntry te = (TransactionEntry)  
    assoc.getEntry(key);  
if (te == null) {  
    te = new TransactionEntry(key, dur);  
    assoc.put(key, te);  
} else {  
    assert te.getState() == te.REMOVED;  
    te.recreate(dur, session);  
}
```

What this code is about doesn't matter. As you can tell just from the control flow, I firmly believe that if `assoc.getEntry(key)` returns a nonnull `TransactionEntry`, then that `TransactionEntry` must be in the `REMOVED` state. This is a desired property of my system and is enforced elsewhere (or so I believe), but is far from obvious in this piece of code. The

parasoft

www.parasoft.com

“A” assertions are a cheap and easy way of building confidence in your code



assertion both documents my belief and confirms it at runtime, making me a little more confident that my system is correct.

The Details

Having seen why assertions are a good idea, let's look at Java's assertion facility in more detail.

The Java language has a new statement, the `assert` statement, which takes one of two forms. The simpler form is the one introduced earlier:

```
assert boolExpr;
```

If the expression evaluates to true, nothing happens, but if it evaluates to false, an `AssertionError` is thrown. The new class `AssertionError` is a subclass of `Error`.

If you want to include an additional information string with the `AssertionError`, provide it after the Boolean expression:

```
assert a.length != b.length:  
    "array lengths not equal";
```

The second expression can in fact be of any type, and will be converted to a string in the usual way.

Assertions are disabled by default in Sun's JVM. You can selectively enable them by class loader, package, or class by calling some new methods of the `ClassLoader` class, or more commonly via command-line arguments when you invoke the Java Virtual Machine. The command-line arguments aren't part of the spec, but in Sun's implementation you would use the `-ea` and `-da` options to enable and disable assertions, respectively. For example, to enable all assertions except for those in the `com.astrel.util` package, you would write

```
java -ea -da:com.astrel.util MyApp
```

If you then wanted to reenoble assertions for the `Heap` class within `com.astrel.util`, you could write

```
java -ea -da:com.astrel.util \  
    -ea:com.astrel.util.Heap MyApp
```

Although the `ClassLoader` methods can be invoked at any time, they'll only affect classes loaded after the call. In other words, the status of a class's assertions – enabled or disabled – is determined once and for all when the class is loaded.

No method will tell you whether the current class has assertions enabled, but you can easily determine this with the following code:

```
boolean enabled = false;  
assert enabled = true;
```

Here the `assert` statement is used only for its side effect. If assertions are enabled, the Boolean will be set to true; otherwise it will remain false.

That's all there is to using assertions. Now some guidelines on using them well.

Tips on Using Assertions

- **Disable assertions for deployment.**

Some people say that assertions should never be disabled. Disabling assertions for deployment, as the saying goes, is like throwing the lifeboats overboard just before the ship leaves port. I agree in principle, but disagree on a technicality. The main benefit of Java assertions is the ability to disable them. If you don't plan on disabling them, you shouldn't be using them. I do believe, and fervently, that your code should perform as much internal checking as is feasible, and that these checks should not be removed. But such checks shouldn't be assertions, in the narrow technical sense of being written with Java's `assert` statement.

For the remainder of this article, I'll use the word "check" to mean any self-validating bit of code that shouldn't be disabled.

- **Make it possible to reenoble assertions at any time.**

Although your deployed program will run with assertions disabled, it should be possible even for the end user to reenoble them when necessary by restarting the application with an appropriate flag. The assertion facility is designed so that assertions can remain in the deployed class file and yet still have no impact on running time – disabled assertions can be removed by the class loader or JIT compiler. (That's why you can't change the assertion status of a class dynamically.) Thus your application need only provide access to the appropriate JVM command-line option in order to enable assertions in the field. By providing this ability, you give yourself another tool for catching bugs in the wild, under conditions that you may not be able to duplicate on your own.

- **Don't use side effects in assertions.**

Since assertions will be disabled in deployed code, they should not change the state of the system – otherwise, the system would behave differently when deployed. (It may behave differently anyway, because disabling an assertion may affect the timing of multithreaded programs, but hopefully your design is robust to withstand such minor timing changes.) There are a couple of techniques that violate this rule: one, which detects whether assertions are enabled, we saw above; another I'll describe later when I discuss postconditions.

- **Assertions should be expensive.**

In other words, either the Boolean expression should take a long time to execute, or the cumulative time of executing the assertion in typical runs of your program should be large. If the assertion doesn't slow you down, why bother disabling it? Or to put it another way: write cheap tests as checks, not assertions. Those checks will help you catch bugs in the deployed system without impacting performance. You can't beat that.

Of course, if assertions as a whole are too expensive, your program will run too slowly to test, and you'll test less frequently. In this way, assertions can actually decrease the robustness of your code. Here is where the ability to selectively disable assertions comes in handy. When a class or package is first coming up to speed, enable assertions in it to gain con-

introware

www.introware.com

fidence in its correctness. When the program becomes too slow to test frequently, disable assertions in the older, well-tested parts of the system.

- **If you can recover from it, don't assert it.**

Assertions throw an error instead of an exception because their purpose is to crash your program. (The whole reason behind having separate Error and Exception classes is that Errors indicate faults from which you shouldn't try to recover.) So catching an AssertionError and trying to continue is a sure sign that you should be doing a check, not an assertion. It's reasonable to catch an AssertionError, but only to log or otherwise process it before your program exits. In other words, any such catch should end by rethrowing the error.

- **Don't mention assertions in documentation.**

An assertion is an implementation choice, like the name of a local variable; whether it's enabled, disabled, or exists at all should in no way affect the contract of a method. (But feel free to boast to your co-workers that you've made your code more robust by "asserting the heck out of it.")

- **Don't use assertions for argument checking.**

If your method's contract claims it will check for null arguments, then you should do just that – check, not assert. There are two problems with using an assertion. First, the assertion will be disabled in production runs of the program, so your method won't be up to spec. (That really is like throwing the lifeboats overboard.) Second, the assert statement throws the wrong kind of thing, an AssertionError. If an argument is null, you should be throwing a NullPointerException; if it's generally invalid, an IllegalArgumentException; and so on.

If you're tempted to use assert anyway because you can write the brief

```
assert arg != null;
```

in place of the verbose

```
if (arg == null)
    throw new NullPointerException();
```

then how about writing a helper method? In fact, let's postulate a class full of them:

```
Check.isNotNull(arg);
```

will throw a NullPointerException, while

```
Check.legalArg(arg.length > 0);
```

would throw IllegalArgumentException. (Two-argument versions of these methods would take message strings to be included in the exception, just like the assert statement.) Concerned about the overhead of an extra method call? Don't be: these Check methods are static and small, which means that a good JIT compiler could easily inline them. (On my sys-

tem, running JDK 1.3 with the HotSpot Client VM, there's no measurable time difference between calling a Check method and writing the same code inline.)

All that being said, if checking an argument is expensive, consider using an assertion instead (and don't claim in the documentation that the argument is checked). Another time to consider an assertion over a check is when you control all calls to the method – for example, when the method is private. Since you can guarantee that all calls to the method are correct, you don't need a check, but an assertion couldn't hurt.

- **Use assertions for preconditions and postconditions.**

A precondition is something your method assumes to be true on entry. Restrictions on arguments are one kind of precondition, but not the only kind: your method might require that other parts of the system be in a certain state before it can validly proceed. Once you've identified a precondition and decided that you don't want to turn it into a check (presumably because it's too expensive), asserting it is a good idea.

One kind of precondition is a class invariant – a statement about a class or its instances that should be true before and after each of the class's methods. For example, say you are writing a Heap class that implements a binary heap – a binary tree with the property that the value stored in each node is no less than the values stored in its children. This property is a class invariant. (Heaps are useful for building priority queues, among other things.)

At the beginning of the method that adds a new item to the heap, you expect that the heap property is true. Indeed, the standard algorithm for adding an item to a binary heap won't work correctly unless it's true. So the heap property is a precondition of the add method. It would be wise to write an isValid method in the Heap class that checks the property, and place

```
assert isValid();
```

at the start of the add method.

Similarly, a postcondition is something that should hold when the method is finished. One seldom checks postconditions, but asserting them makes sense. Class invariants are postconditions as well as preconditions, so it would be a good idea to assert the heap property at the end of each Heap method as well as at the beginning.

If a method changes its parameters, chances are the method's postcondition will need access to the original parameter values. You can write the postcondition by storing copies of the original values before they're modified. For example, the postcondition of any sort routine is that the input array be in sorted order, and that it consist of the same elements as before the sort. While the first condition does not require the original array, the second one does.

Copying parameters is expensive, so it should happen only when assertions are enabled. You can achieve that by using a side effect in an assertion:

```
void sort(int[] a) {
    int[] old_a = null;
```

The main benefit of Java assertions is the ability to disable them. If you don't plan on disabling them, you shouldn't be using them



ashnasoft

www.ashnasoft.com

ASSERTION DOS AND DON'TS

- Do disable assertions for deployment, but make it easy to reenable them in the field.
- Do use assertions for expensive tests – otherwise, use a nonremovable check.
- Do use assertions for preconditions and postconditions.
- Do place assertions in the middle of methods, not just at the beginning and end.
- Do think of assertions as executable documentation.
- Do think of assertions as tests.
- Do write assertions as you write your code.
- Don't use side effects in assertions.
- Don't try to recover from an assertion – let it crash your program.
- Don't mention assertions in documentation – they are an implementation detail.
- Don't use assertions to check arguments.
- Don't use assertions to mark unreachable code

```
assert (old_a = (int[]) a.clone()) != null;
// do the sort
assert inSortedOrder(a) &&
    hasSameElements(a, old_a);
}
```

The inequality check in the first assertion is there only to produce a Boolean that is always true; the assertion's real job

is to late a `Check.impossible()` method that throws the exception and write:

```
switch (x) {
    case 1: ...; break;
    case 2: ...; break;
    default: Check.impossible();
}
```

- **Use assertions as executable documentation.**

Occasionally you may get the impulse to write a brief comment in the middle of your code to the effect that, for example:

```
// at this point, x is greater than y
```

Instead of commenting it, assert it:

```
assert x > y;
```

The assertion is as valuable as the comment to the human reader, and it's machine-checkable as well.

- **Treat assertions as tests.**

Assertions are the first line of testing – they are subunit tests. Once you realize this, many of the above guidelines fall

Like a well-written test, a well-placed assertion will catch bugs early in the development process

is to clone the array. If assertions are disabled, no copying will occur.

- **Use assertions for those hard-to-reach places in the middle.**

Assertions are great for places in the middle of methods when part of the job is done. An assertion in the middle of a method will catch problems sooner than one at the end, and may have less work to do as well. For instance, part of the process of adding a new item to a heap involves repeatedly comparing a node's value in the tree with those of its two children, and then possibly swapping the node with its larger child. Placing an assertion after this piece of code, to the effect that the heap property still holds for the parent node and its children is cheaper than testing the whole heap for validity at the end, will catch any problems in the code right where they happen, and acts as useful documentation too.

- **Don't use assertions to flag "unreachable" code.**

We often write switch statements whose default case we know cannot happen, and sometimes the compiler forces us to catch exceptions that we know cannot be thrown. Some suggest using assertions in these and other allegedly impossible-to-reach places, but I disagree. Checks are preferred: they will not be disabled and, by assumption, will not affect performance, since they will never run. Simply writing a throw statement would be adequate (although exactly which class of exception should be thrown is not clear), or we could postu-

out almost automatically. You disable assertions in production for the same reasons that you don't ship your test code. Asserting the postcondition of a public method is much like writing a unit test for that method. And the importance of midmethod assertions is clear: they're located in places where you can't otherwise test.

- **Assert early and often.**

Write assertions as you write your code – or even before, if you're a devotee of Extreme Programming. That is when the logic's details are freshest in your mind. And feel free to write plenty of them, since you can always disable them.

Conclusion

Although assertions are a small addition to Java – just one new statement – they can have an impact out of proportion to their size. Like a well-written test, a well-placed assertion will catch bugs early in the development process, when it's cheapest to fix them. They serve as helpful documentation to readers of your code. And since assertions can be disabled at runtime, your program's performance need not suffer. So don't hesitate to use them. Assertions are sure to improve the quality of your code. 🍎

AUTHOR BIO

Jonathan Amsterdam is a senior consulting engineer at DataSynapse, Inc. Formerly, he was president and founder of Astrel, Inc., a Java training and consulting firm. He's also an adjunct professor of computer science at New York University.

amsterdam@cs.nyu.edu

new atlanta
www.newatlanta.com

Synchronizing Java Threads on a Shared Resource with Multiple Views

A simple and elegant solution



WRITTEN BY
VISHAL GOENKA

Java thread synchronization primitives are based on object instances. Multithreaded access to a shared resource requires a unique object instance that all threads accessing the resource can synchronize upon.

This is especially challenging for resources that may have multiple views. For instance, multiple threads can independently open a given file and will have separate instances of the `java.io.File` object, each corresponding to the same file. The different object instances that correspond to multiple views of the same resources don't allow synchronized multithreaded access to these resources.

This article illustrates the problem and examines approaches to solving it with an emphasis on their synchronization and concurrency trade-offs. It presents a few use-case examples where this problem manifests itself, followed by a simple and elegant solution with the complete Java source code. Java programmers who work with remote resources, including file, URL, LDAP, JNDI, and relational databases, are likely to find this article helpful in recognizing

ing areas of code that are vulnerable to suboptimal synchronization.

How Java Thread Synchronization Works

Each Java object has a "monitor" that can be used as a semaphore to synchronize multithreaded access to shared resources. Typical resources that are shared in a Java program include shared memory spaces such as tables, queues, and lists. Various Java classes such as `java.util.Vector` and `java.util.Hashtable` have most of their methods synchronized on the object instance. As you may already know, any Java object may be used as the semaphore for synchronizing multithreaded access to itself or other object(s), as long as each thread uses the same instance of the object to synchronize upon.

The Challenge with Multiview Resources

Shared system and network resources, such as files, database tables, network connections, URLs, and LDAP directory entries, are also represented by one or more Java objects, and often the same underlying resource is represented by more than one object. Each object that represents a resource presents a view of that resource. While it's sometimes possible and desirable to use a unique object instance that corresponds to a given underlying resource, in many cases it's either undesirable or impossible to do so. Figure 1 illustrates the synchronization and concurrency challenges posed by multiple views of a given resource.

Consider an application that needs to write some data to one of several hundred files in a given file system. The file is selected based on the request

parameters. Assume that a multithreaded request dispatcher handles each incoming request, parses the request to map it to a unique file in the file system, and then edits the file as per the request parameters. Given the possibility that multiple simultaneous requests can map to the same file, the application must ensure that access to a given file is synchronized across multiple threads.

The implementation in Listing 1 is clearly incorrect, since the synchronization is based on a local object instance. Two threads processing independent requests that return the same file name will create two separate `java.io.File` objects. Therefore no synchronization will be achieved. Making the entire method synchronized introduces more problems than it solves. First, it significantly reduces concurrency by synchronizing access across independent files. Second, it doesn't help if the request can be dispatched to one of many methods, each of which may need to access the requested file. Synchronizing every method that may need to access any file is clearly unacceptable, as it severely reduces concurrency.

First Cut at the Solution

Our first attempt at solving this problem is to make use of a shared table that keeps track of the files in use and returns the same instance of a `File` object to every request that corresponds to a given file. As a request is made for a file, the table is checked to see if a `File` object corresponding to the requested file already exists. If so, the existing `File` object is returned and the use count for the `File` object is incremented by one. Otherwise, a new `File` object is created

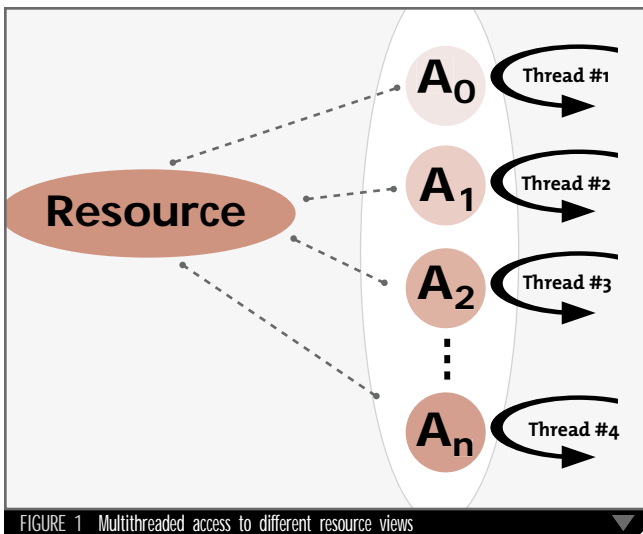


FIGURE 1 Multithreaded access to different resource views

visual mining

www.visualmining.com

for the file and added to the table with the use count set to one.

Once the requester is done with the file, it must explicitly call a method that decrements the use count by one and removes the File object from the shared table if the use count is reduced to zero. Listing 2 provides the modified code based on this solution. The method `acquireFile(String)` is used to request the object instance for a file and increment the use count; the method `releaseFile(File)` is used to decrement the use count and release the File object if it's no longer in use.

Critical Analysis

Note: The try-finally block becomes necessary to ensure that the file use count is decremented even if the method throws an undeclared throwable in the synchronized block. A missed finally block can cause undesirable side effects such as memory leaks, making this approach unattractive. This approach is also unacceptable if the File object is to maintain a state that is specific to each thread requesting access to the file. The limitation is easy to get around once we realize that there are two distinct requirements here:

1. To obtain a reference to a File object corresponding to the requested filename
2. To obtain a unique object to synchronize the file access on

In the solution in Listing 2 we returned the same object for both purposes, whereas we could just as well return a different object for synchronization.

Second Attempt at the Solution

We modify the solution (see Listing 3) to create a local instance of the File object and seek a unique object instance

for synchronization purposes only. The method `acquireSemaphore(File)` checks if the given File instance is equal to any other File that exists in its tables. If so, it returns the unique object stored against the existing File entry and increments the use count. Otherwise, it stores the current File instance in its tables along with a new `java.lang.Object` instance created to act as the semaphore and sets the use count to one. The method `releaseSemaphore(File)` decrements the use count for the File that's equal to the given File, and clears the entry if the use count goes to zero. Using a `java.util.Hashtable` (or `java.util.HashMap`) makes the equality comparison easy and efficient as long as the `hashCode` returned is also the same for objects that are equal (though not necessarily the same instance).

A Few Use Cases

Before looking at further improvements to the solution developed thus far, let's discuss a couple more use cases so that we can arrive at a clear definition of the problem.

First, consider a directory-based application that uses LDAP or JNDI APIs to access the directory. For optimal concurrency it may be desirable to synchronize thread access at the level of a directory entry. A directory entry can be uniquely identified using a canonical string representation of its "DN" or distinguished name. Using the solution developed above, an application fragment may be written as illustrated in Listing 4 to allow for maximum concurrency while remaining thread-safe.

Next, we have a server application that caters to authenticated users and needs to limit access to certain resources to one concurrent request per user. The solution outlined in Listing 4

can be applied to all such resources by using the unique identity of the authenticated user to obtain a semaphore for synchronization.

Formal Definition of the Problem

As you may notice, for the `acquireSemaphore` method to work correctly in these examples, the argument must uniquely represent the shared resource and must also be equal to other argument instances that represent other views of the same resource. The problem can therefore be described as a requirement to synchronize across n objects that are not equal by reference, but are equal by value. In other words, these n objects are different views of the same resource.

To avoid confusion, in the following description I'll use the term *equal* to imply reference equality, and *equivalent* to imply equality by value. Thus equal objects are always equivalent, though not the other way round. Therefore, these n objects are equivalent but not equal to each other. Expressed in Java, it implies that the following is true for any given n objects `obj[0]` through `obj[n-1]` that we need to synchronize across:

```
obj[i] != obj[j]
// for all i != j, 0 <= i < n,
    0 <= j < n
• obj[i].equals(obj[j])
• (obj[i].hashCode() ==
  obj[j].hashCode())
// for all i, j, 0 <= i < n, 0
  <= j < n
```

The solution is simple and elegant and is illustrated in its entirety in Listing 5. Listing 6 presents Listing 3, rewritten using the solution.

How Does the Solution Work?

Start with any object that satisfies the synchronization requirements listed above. Instead of synchronizing on the object itself, obtain a semaphore for the object using a single instance of the monitor that's accessible to all concerned packages. The monitor maintains weak references to all objects with semaphores in a hash map for faster lookup. The weak reference allows the objects to be garbage collected if there are no other strong references to the object, obviating the need for maintaining use counts using try-finally blocks.

When you request a semaphore for an object, the monitor looks for an object in its hash table that's equivalent to the given object (has the same hash code and satisfies the equals method). If an equivalent object is found, the same is returned.

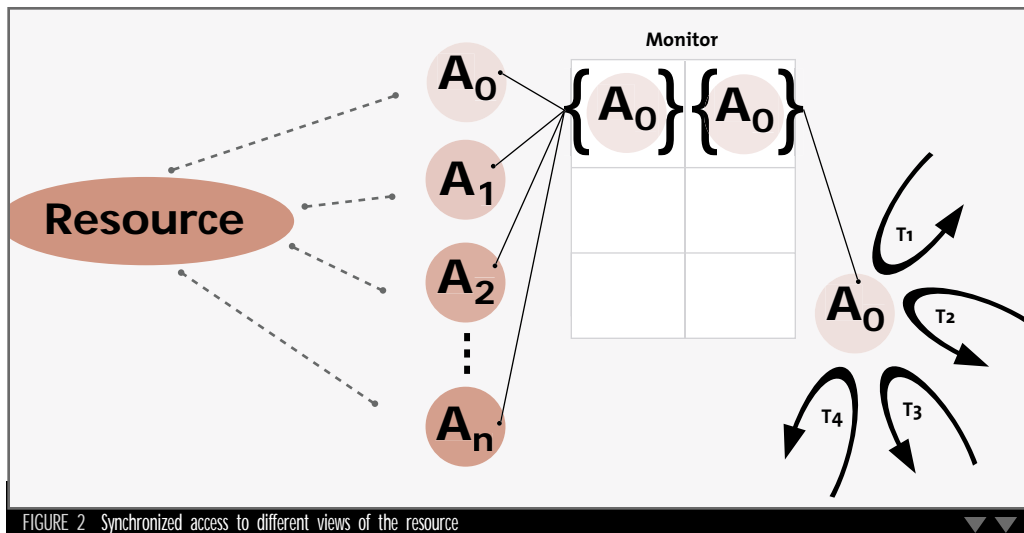


FIGURE 2 Synchronized access to different views of the resource

ilog
www.ilog.com

Otherwise, the given object is added to the hash table, wrapped in a weak reference so that the garbage collector automatically removes it when it's no longer in use.

This approach always keeps one element of every equivalence set in the hash table until no element in that equivalence set is in use, at which point it's eligible to be removed until the next time it's required. Therefore, it's best to obtain a semaphore for a lightweight object that's a simple canonical representation of the resource.

Unlike the example in Listing 2, the semaphore (which is really the first object in its equivalence set to look for a semaphore) is used only for synchronization, not for any other access, thereby allowing context-specific settings to be different among different objects in the same equivalence set. Wrapping the semaphore in a weak reference eliminates the need for maintaining use counts and makes the usage more natural. Figure 2 demonstrates the solution.

The Nuts and Bolts

To conclude, let's take a closer look at the code in the monitor. The following statement in the monitor may also return a null in the case when ref is non-null.

```
Object monitor = ((ref == null) ?
null : ((WeakReference)ref).get());
```

This is possible in the case of an unlikely race condition in which the object is cleared from the weak reference after the `super.get(key)` statement, at which point it may still be in the hash table. Checking for `monitor == null` again guards against a rare race condition. The `get` and `put` methods of the underlying `HashMap` (a `WeakHashMap`) are not synchronized; this is good since we don't expose them directly but through the single synchronized `get` method in the monitor. Synchronizing the `get` method of the monitor is essential, as it involves modification of a shared data structure.

While a single instance of the monitor may be functionally sufficient for the entire application, for higher concurrency a different instance of monitor should be used for independent modules or packages that require semaphores for objects of different classes. The monitor tremendously simplifies the writing of multithreaded code with just the right amount of synchronization – not more, not less. ☛

▼▼▼ vgoenka@campuspipeline.com

Listing 1

```
public void process (Request req) {
    File file = new File(req.getFileName());
    synchronized (file) {
        // ... open/modify/close file as per the request parameters...
    }
}
```

Listing 2

```
public void process (Request req) {
    File file = acquireFile( req.getFileName() );
    try {
        synchronized (file) {
            // ... open/modify/close file as per the request parameters...
        }
    } finally {
        releaseFile( file );
    }
}
```

Listing 3

```
public void process (Request req) {
    File file = new File( req.getFileName() );
    Object semaphore = acquireSemaphore ( file );
    try {
        synchronized (semaphore) {
            // ... open/modify/close file as per the request parameters...
        }
    } finally {
        releaseSemaphore( file );
    }
}
```

Listing 4

```
public void methodA(LDAPObject obj) {
    String dn = obj.getDN();
    Object semaphore = acquireSemaphore ( dn );
    try {
        synchronized ( semaphore ) {
            // read/modify/delete object
        }
    } finally {
        releaseSemaphore ( dn );
    }
}
```

Listing 5

```
public class Monitor {
    private WeakHashMap map = new WeakHashMap() {
        public final Object get(Object key) {
            Object ref = super.get(key);
            Object monitor = ((ref == null) ? null :
((WeakReference)ref).get());
            if (monitor == null) {
                monitor = key;
                put (monitor, new WeakReference(monitor));
            }
            return monitor;
        }
    };
    public synchronized Object get(Object key) {
        return map.get(key);
    }
}
```

Listing 6

```
static final Monitor monitor = new Monitor();

public void process (Request req) {
    File file = new File( req.getFileName() );
    synchronized (monitor.get(file)) {
        // ... open/modify/close file as per the request parameters...
    }
}
```

▼▼▼ Download the Code!
www.javadevelopersjournal.com



J2ME



J2SE



J2EE



Home

AUTHOR BIO

Vishal Goenka is a senior software engineer for Campus Pipeline Inc., where his group is responsible for the infrastructure components of the Campus Pipeline Web Platform. He focuses on security architecture and implementation. Vishal holds a BS in computer science from the Indian Institute of Technology Kanpur (India).

actuate

www.actuate.com

Evolutionary Java Applications



WRITTEN BY
ALVARO
SCHWARZBERG

For the past few years I've participated in several projects to update existing Java applications. While working on those projects I often wanted to be able to add new functionality to a class without recompiling it.

Some of the reasons for this were:

- I didn't have the right to access or modify a class's source file if, for example, the class was developed by a tier supplier.
- The class was used in many distributed sites and the new functionality was useful for a small number of sites only.
- There were some persistent objects and I didn't want to manage several versions of the class or update all the persistent objects using a batch mechanism.
- I didn't want to touch a tested class that had been working well for many months or years.
- Other developers were working on the same class at the same time and we had to coordinate our modifications to avoid any incompatibility as well as avoid scratching our modifications. It would have been a lot easier if each developer had worked on a different source file associated with its own functionality.

After much thought I decided I didn't want a completely dynamic solution, such as JavaScript, where it's possible to add new functions to a class at runtime. In that kind of language the performance is affected too much and the application architecture becomes difficult to master.

Finally, I created Dynamic Java Binder (DJBinder), a tool that enables you to dynamically attach interface implementations to a class without changing its source file. The dynamically attached interfaces are equivalent to the interfaces listed after the "implements" keywords.

DJBinder uses the class loading mechanism of the Java 2 platform to create the link between the classes and the interfaces at runtime.

How DJBinder Works

An Example Application

I have an application that creates objects of type Man and Woman. Now it's time to add a new print functionality that writes the name and age of each person to the standard output. To reduce the regression risk, the solution can't modify the existing application code.

Listing 1 shows all the definitions used in this example. *Note:* These classes have not been designed with the intention of using any dynamic mechanism.

First I'll present the Java code you need to write to solve this problem using DJBinder, then I'll explain how DJBinder handles the internal details.

The following statements write the person "p" data to the standard output using the Print interface:

```
Person p = ... ;
((Print) p).toStandardOutput ();
```

The cast operation is needed because the Print interface isn't directly implemented by the Person class. The Java compiler assumes that the Print interface is implemented by a Person subclass, but we know this isn't true since neither Man nor Woman implements the Print interface.

The DJBinder method that implements the Print interface without changing the Person class is a new abstract class named DI_Person_Print:

```
public abstract class
DI_Person_Print implements Print
{
    public void toStandardOutput() {
        Person p= (Person) (Object) this;
        System.out.println (p.getName());
    }
}
```

Enable functionality-driven software architecture

The class name is very important because it creates the link between the Person class and the Print interface. The class is abstract since it can't be directly instantiated; its instances are implicitly created by DJBinder according to the cast operations.

The main line of the toStandardOutput() method is:

```
Person p= (Person) (Object)this;
```

This enables you to get the Person object that's associated with this interface implementation. The cast isn't done directly because the Java compiler forbids a cast operation between two classes that don't belong to the same hierarchy. The trick is to use an Object class, such as bridge, which is legal because the Object class is the root of all the class hierarchies.

Writing the age is a little more difficult because the age member has a protected visibility that forbids access from the DI_Person_Print class.

The solution proposed by DJBinder is to create a new class with all the protected and private members of the Person class that may be accessed from the DI_Person_* classes. The name of this class must be DA_Person. For example, the following class authorizes access to the age member:

```
abstract class DA_Person{
    public int age;
}
```

This class makes it possible to use the age member in the toStandardOutput() method:

```
public abstract class
DI_Person_Print implements Print
{
    public void toStandardOutput() {
        Person p = (Person) (Object)
```

preemptive

www.preemptive.com

```

    this ;
    System.out.println
        (p.getName()) ;

    DA_Person pp = (DA_Person)
        (Object) this;
    System.out.println (pp.age) ;
}
}

```

Notice that the object referred to by the pseudo variable “this” is cast to the DA_Person class to access the protected member age. An equivalent alternative would be to cast the variable “p” to the DA_Person class :

```

DA_Person pp = (DA_Person)
    (Object)p;

```

The final step of my example is to loop over all the persons created by the existing application and cast each person to the Print interface (see Listing 2).

The mechanism implemented by DJBinder is similar to the inner classes of Java. The DI_* classes are similar to inner classes. The most important difference is that each DI_* class is defined in a different source file and can belong to a different JAR file.

The fact that each DI_* class can be extended from another class offers an

elegant way to implement the multiple heritage in Java without the problems of other languages such as C++.

The code shown in Listing 2 compiles well, but at runtime there would be several errors if DJBinder didn't use the class loading mechanism to change the bytecode on the fly.

The class loading mechanism works in the following way: every time the Java Virtual Machine needs a new class, it requests the current class loader to return the corresponding bytecode. For example, the default class loader returns the content of a file named class-Name.class. This file is searched in the directories listed in the CLASSPATH environment variable.

The DJBinder class loader works in a different way. First, it gets the original bytecode using the class loader that's normally used by the application, changes it, then returns the modified bytecode to the virtual machine.

The DJBinder tool requires that the JVM uses the DJBinder class loader. For the console programs you simply need to replace the traditional command:

```

virtualMachinePath javaParameters
applicationName
applicationParameters

```

with

```

virtualMachinePath
javaParameters
amslib.djbinder.Start
applicationName
applicationParameters

```

The amslib.djbinder.Start class runs your application using the DJBinder class loader.

For other kinds of Java programs (applets, servlets, JSPs, etc.) there's a similar procedure based on the runtime environment characteristics. For example, if you run your Java application server using the amslib.djbinder.Start class, the beans and servlets become DJBinder aware and can use any interface that's dynamically implemented.

The DJBinder class loader changes the original bytecode in four places:

1. The cast operations with an interface of target type

The exceptions eventually thrown by the cast operations are caught and DJBinder tries

to find a class named DI_objectClass_interface or DI_objectSuperClass_interface. If one correct class is found, a dynamic interface implementation object (DI_* object) is created and returned, otherwise the original exception is rethrown. The objectSuperClass token represents any class in the hierarchy up to the java.lang.Object.

If the same cast is done twice for the same object, the previous DI_* object is returned instead of creating a new object. This allows you to keep some information within the DI_* objects.

In my example this bytecode change prevents the ClassCastException that should be thrown by the following statement of the PrintAllPersons class (see Listing 2):

```

Print d= (Print)e.nextElement();

```

The object returned by the expression e.nextElement() is an instance of Person; normally it can't be cast to the Print interface, but DJBinder creates the corresponding DI_* object, which becomes the result of the cast operation.

2. The cast operations that have a class as target type

If the casted object is a dynamic interface implementation object, DJBinder replaces the cast by a reference to the main object, that is, the object that triggered the creation of the DI_* object.

In my example this bytecode change prevents the exceptions that should be thrown by the statements:

```

Person p= (Person) (Object)this;
DA_Person pp =
    (DA_Person) (Object) this;

```

3. The references to a DA_* class

The DA_* classes are a buildtime artifice and don't exist in the runtime environment. All the references to a DA_XXX class are converted into a reference to the XXX class. In addition, special methods are added to the XXX class to enable controlled access to the private and protected members. The access is allowed from the DI_XXX_* classes only.

In my example this bytecode change enables you to access the protected age member of the Person class.

4. The instanceof and == operations

These operations are modified to get a behavior that's compatible with the result of the cast operation: if a cast operation throws an exception, the cor-

north wood p/u

AUTHOR BIO

Alvaro Schwarzberg works in software development for Dassault Systemes, and other international companies based in Colombia, Brazil, and the U.S. He has experience working with multilayered object-oriented applications mixing Java, C++, and databases.

capeclear

www.capeclear.com

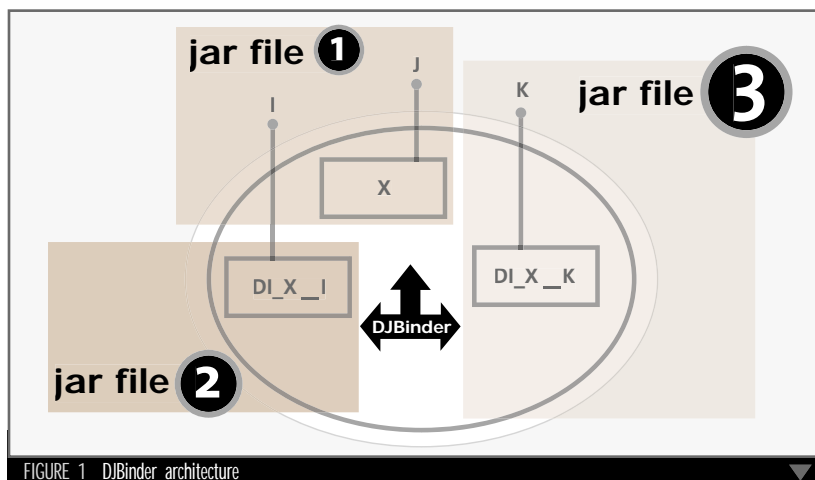


FIGURE 1 DJBinder architecture

Listing 1: Definitions used in the example

```

class Person
{
    public Person (String n, int a)
    {
        name=n; age=a; allPersons.add(this);
    };

    public static Enumeration getAll()
    {
        return allPersons.elements();
    };

    public String getName()
    {
        return name;
    };

    protected static Vector allPersons =
        new
        Vector();
    protected String name ;
    protected int age ;
}

public class Man extends Person
{
    public Man (String n, int a) {super(n, a);}
}

public class Woman extends Person
{
    public Woman (String n, int a) {super(n, a);}
}

public interface Print
{
    void toStandardOutput() ;
}

```

Listing 2: Full solution public class

```

PrintAllPersons
{
    public static void main (String [] arg)
    {
        Enumeration e = Person.getAll();
        while (e.hasMoreElements())
        {
            Print d = (Print) e.nextElement() ;
            d.toStandardOutput () ;
        }
    }
}

abstract class DA_Person
{
    public int age;
}

public abstract class DI_Person_Print
implements Print
{
    public void toStandardOutput()
    {
        Person p = (Person) (Object) this ;
        System.out.println (p.getName()) ;

        DA_Person pp = (DA_Person) (Object) this ;
        System.out.println (pp.age) ;
    }
}

```

Download the Code!
www.javadevelopersjournal.com

responding instanceof operation must return false; if a cast operation works silently, the corresponding instanceof operation must return true.

Let's assume that the classes X, DI_X_I, and DI_X_J exist and the "v" variable refers to an object of type X, therefore the cast operations:

```
(I) v
(J) v
```

do not throw an exception, and the operations:

```
v instanceof I
v == (I) v
(J) v == (I) v
```

return "true."

These changes create the illusion that the main object and the dynamic interface implementation objects are a single object. This simple fact makes programming a lot easier. Figure 1 shows the software architecture allowed by DJBinder.

A final requirement of DJBinder is that the DI_* and DA_* classes must belong to the same package of the main class or to a subpackage named djbinder. For example, if the Person class belongs to the acme.applis package, the DI_Person_Print and DA_Person classes must belong to the acme.applis package or the acme.applis.djbinder package.

The djbinder subpackage allows you to use the DJBinder mechanism for classes that belong to a sealed package – a package that can't be modified.

What do software developers and application administrators gain from using DJBinder?

For developers the modification unit is usually the source file. Each source file has a creation and modification date. In large software projects with several

developers it's necessary to establish a reservation mechanism to prevent two developers from changing the same file simultaneously, otherwise one of the modifications would be lost. DJBinder allows you to distribute the tasks and responsibilities among the developers better, so each developer can work on a well-defined set of functionalities (grouped within an interface). This feature facilitates the parallel work of several developers (increasing the concurrent engineering), thus reducing the duration of the project.

For customers and administrators the modification unit is the executable file. In a Java application the executable files are the JAR files. DJBinder allows you to build JAR files associated with each functional modification. New functionalities become available when the corresponding JAR file is added to the runtime environment. The JAR files already delivered with the previous versions of the application don't have to be modified. This feature facilitates the administration and reassures customers who are afraid of regressions.

A feature that can be easily designed using DJBinder is virtual typing – using a type that's different from the Java class name to look for a dynamic interface implementation. For example, you could assign the virtual type "French" to an instance of the Person class. In that case the Print interface could be implemented by the DI_French_Print class and include some extra fields. Different instances of the same Person class could have different virtual types. This extra flexibility is very useful, especially to communicate with legacy applications or legacy databases.

Conclusion

DJBinder enables a functionality-driven software architecture that's particularly flexible and makes the evolution of any Java application easier, including existing applications, without recompiling them. DJBinder uses the Java class loading mechanism and doesn't need any special compiler or virtual machine.

For more information about the class loading mechanism see <http://java.sun.com/products/jdk/1.2/docs/api/java/lang/ClassLoader.html>, or contact me at alvaro.schwarzberg@amslib.com.

You can download a full version of DJBinder from www.amslib.com/djbinder; however, this version can't be used for commercial development without my written authorization. ☉

alvaro.schwarzberg@free.fr

pointbase

www.pointbase.com



JASON BRIGGS J2ME EDITOR

A Perfect World

I was reading a forum discussion recently that argued that J2ME was a mess. The general consensus (admittedly there weren't that many messages) seemed to be that this conclusion was correct. My automatic response was "What a complete load of bollocks" (which I think means I've been living in England far too long). However, upon reflection, I still don't entirely agree, but I don't disagree either. I am officially in-between camps. Neutral. Unbiased (yeah, right).

If you mistakenly view J2ME as the sum of all Java technologies for embedded and mobile devices, then yes, it can be confusing. You have a couple of flavors of Waba, which have nothing to do with Sun and aren't part of J2ME at all. There's also iAppli, NTT DoCoMo's "somewhat-similar-to-MIDP-but-not-quite" Java API for their i-mode mobile phones. You have PersonalJava, which is, kind of, part of the J2ME family, but was around first so it doesn't really fit. And you have JavaPhone, Java TV, and EmbeddedJava. Then we launch into buzzword territory: MIDP, CLDC, CDC, KVM, CVM...I could probably keep going. It's hardly surprising that, looking at this bewildering array of products and APIs, people turn somewhat chalky gray and decide there's no sense of direction there.

For a moment, imagine the world is a better place. No one has taken potshots at each other since World War II, John Lennon was never assassinated, the Beatles never broke up and now head a global chain of McBeatles vegetarian fast food and music stores. I won the lottery last year and am consequently living a life of indolent luxury somewhere on my own private Pacific island (with a luxury yacht, don't forget the yacht).

And no one worries about legacy systems.

"What, our financial application was written a hundred years ago in Cobol?"

Don't worry about it. Scrap the whole thing and rewrite it in Java!"

In this perfect world (my perfect, not necessarily yours), J2ME becomes the fundamental structure of the entire Java platform. Configurations define the base (non-GUI) level of an API, and each configuration builds upon the ones below it. Profiles provide the user interface API with a product and, if they don't build up from one another in a hierarchical relationship like configurations, at the very least they share large chunks of their API set. CDC inherits from CLDC. The Personal Profile (goodbye PersonalJava) builds on top of CDC and MIDP builds on top of CLDC.

Developers can take advantage of this structure when writing applications. An application is partitioned, separating the user interface from the rest of the code. A certain amount of rewriting or additional coding will always be required when, for example, moving an application from a mobile phone up to a PDA, simply because of the variation in device and functionality provided; however, the amount of rework will hopefully be limited by the basic design.

The funny thing is, carried to the logical (and yes, a bit oversimplified) conclusion, this structure could propagate up into J2SE and J2EE with each inheriting a configuration and then defining a profile for the "extra bits." Which means J2ME becomes the central core of Java - **JDJ** gets renamed **J2ME Developer's Journal**, and I reign supreme from my tropical resort: "Bow before me Alan, Ajit, and Jeremy!"

Ahem.

Alas, the world is a slightly more complicated place. The various JCP Expert Groups can't apply a rule of thumb that says we can ignore what has gone before and do things the way they should be done. So changes have to trickle through, slowly (hands up - who's still waiting for the Personal Profile to make an appearance?).

A Perfect World

I was reading a forum discussion recently that argued that J2ME was a mess. The general consensus (admittedly there weren't that many messages) seemed to be that this conclusion was correct. My automatic response was "What a complete load of bollocks."

by Jason Briggs

72

When Should I Use JMS?

Three alternatives to JMS

by Nigel Thomas

74

J2ME Benchmarking: A Review

Evaluating an application's performance objectively

by Carl Barratt and Glenn Coates

78

Hardware Accelerators for J2ME Come of Age

Java gets a needed boost

by Ron Stein

84

For the moment, we're stuck with the current state of play. If you're developing for mobile phones, MIDP is probably the best choice; for a PDA, use PersonalJava. And ne'er the twain shall meet.

As a result, J2ME winds up looking like a mess of only - passingly - related products.

Still, nothing is perfect. More's the pity.

In this month's **J2ME Developer's Journal**, I mean **Java Developer's Journal**, Ron Stein discusses J2ME hardware accelerators (unfortunately, not quite a bolt-a-box-on-the-side-of-your-processor-and-things-will-go-faster concept). And Glenn Coates and Carl Barratt look at the issues in J2ME benchmarking.

Read on for your monthly fix of J2ME confusion. ☛

▼▼▼ jasonbriggs@sys-con.com

AUTHOR BIO

Jason Briggs is a Java analyst programmer and - sometimes - architect. He's been officially developing in Java for almost four years "unofficially for five."

install shield

www.installshield.com



WRITTEN BY NIGEL THOMAS

When Should I Use JMS?

JMS has been a godsend to Java developers who want to use tried-and-tested messaging paradigms without having to wrestle with multiple proprietary APIs. A new breed of messaging vendors is delivering enterprise-quality JMS implementations at substantially lower costs than the previous MOM incumbents, as well as offering JMS wrappers to help integrate legacy and Java environments and extending JMS to lightweight and mobile devices.

However, JMS is not the only show in town. This article discusses when you might prefer to use three existing alternatives to JMS.

Use Messenger to Simplify JMS Development

For an application developer the complex threading model in JMS can be hard to use; you have to understand which thread owns which session (and consumer and producer), from which session you can send/receive or add a listener, and so on. This can be especially hard in servlet or EJB containers when you don't know what thread your code is called from.

Messenger is a simple open-source framework for working with JMS that takes care of all these complex thread-pooling issues and quality of service configuration options. Developers using Messenger can simply send, receive, or add listeners – then Messenger takes care of the details. The same Messenger instance can be shared across many threads.

At deployment time the exact quality of service and network topology (What JMS provider should be used? Is it XA? What's the delivery mode? Acknowledgment mode? Is it a topic or queue? Is it durable? What's the client name? etc.) is all contained in a single XML deployment document. You can find examples and more information at <http://jakarta.apache.org/commons/messenger.html>.

Messenger is already in use in SpiritSoft's new SpiritCache product; it

helped us simplify the JMS code immensely.

Use JAXM for Interbusiness Messaging

JMS implementations from different vendors are not required to interoperate: details of wire format and transport protocol are left to the provider's discretion. So JMS usage is limited to "single provider" situations, in which the developer controls both producers and consumers.

When the other end of the conversation is another division or company, JMS gives way to wire-format, standard-based APIs, such as JAXM – the Java APIs for XML Messaging – developed by the Java Community Process as JSR 67.

JAXM is designed to support interbusiness messaging. Based on SOAP, message formats, payloads, and transport are standardized so that the other business will be able to receive and understand them. Unlike JMS, JAXM supports only the point-to-point messaging domain – not least because B2B messaging is usually one to one. The high performance, low latency, and programmatic flexibility of JMS are traded for simplicity, reliability, and interoperability. JMS messages can of course be bridged across JAXM.

Use JavaMail for Messages Anywhere

E-mail is slower still than B2B messaging but has a huge installed base – even home users have access to Yahoo! or Hotmail. Just as in the past Telex was used as a hybrid human/machine readable network – you can see its echoes in EDI formats like SWIFT – the JavaMail API is ideal for low-priority, unpredictable messaging where the receiver may be a human or a process, lengthy delivery delays and even message loss are possible, and disconnected clients are likely. Like SOAP, JavaMail supports multipart MIME messages.



Summary

Pick the API (or set of APIs) that best suits your business needs:

- **JMS** is the ideal high-performance messaging platform for intrabusiness messaging, with full programmatic control over quality of service and delivery options.
- **Messenger** provides a simple facade to JMS which – trading flexibility for simplicity – eases development and abstracts complexities into a simple configuration file.
- **JAXM** provides an interbusiness messaging API, supporting complex (but XML only) formats across standard transport protocols, and offering a standard approach for bridging between different JMS providers.
- **JavaMail** provides lowest common denominator, slow, but human-readable messaging using infrastructure already available on virtually every computing platform.

If there's any likelihood that your quality of service or connectivity requirements will change over time, use configurable facades like Messenger to make sure you can easily plug the right solution into your application without having to make expensive code changes. ☛

nigel.thomas@spirit-soft.com

AUTHOR BIO

Nigel is director of product management at SpiritSoft with over 20 years' experience in the industry, specializing in distributed systems architecture and audit.

fiorano
www.fiorano.com



Some of the more commonly asked questions on the various forums for J2ME seem to be, "What is J2ME?" and "Is <so-and-so-product> a part of J2ME?" Here is where you will find all the APIs that fall beneath J2ME's umbrella, and the packages you will find within those APIs.

CONNECTED, LIMITED DEVICE CONFIGURATION (CLDC) – VERSION 1.0

| | |
|-----------------------|--|
| java.io | input and output through data streams |
| java.lang | fundamental classes |
| java.util | collections, data and time facilities, other utilities |
| javax.microedition.io | generic connections classes |

You can find more information on CLDC at the following URL:
<http://java.sun.com/products/cldc/>

CONNECTED DEVICE CONFIGURATION (CDC) – VERSION 0.2

| | |
|-----------------------|---|
| java.io | input and output |
| java.lang | fundamental classes |
| java.lang.ref | reference object classes |
| java.lang.reflect | reflective information about classes |
| java.math | BigInteger support |
| java.net | networking support |
| java.security | security framework |
| java.security.cert | parsing and management of certificates |
| java.text | used for handling text, dates, numbers and messages |
| java.text.resources | contains a base class for locale elements |
| java.util | collections, date/time, miscellaneous functions |
| java.util.jar | reading Jar files |
| java.util.zip | reading Zip files |
| javax.microedition.io | connections classes |

Look for more CDC information here:
<http://java.sun.com/products/cdc/>

MOBILE INFORMATION DEVICE PROFILE – VERSION 1.0

| | |
|---------------------------|---|
| java.io | |
| java.lang | CLDC, plus an additional exception |
| java.util | CLDC, plus timer facilities |
| javax.microedition.io | networking support based upon the CLDC framework |
| javax.microedition.lcdui | for user interfaces for MIDP applications |
| javax.microedition.rms | persistent data storage |
| javax.microedition.midlet | defines applications and interactions between app and environment |

The products page for MIDP is here:
<http://java.sun.com/products/midp/>

FOUNDATION PROFILE – VERSION 0.2

| | |
|--------------------------|--|
| java.io | see CDC |
| java.lang | see CDC |
| java.lang.ref | see CDC |
| java.lang.reflect | see CDC |
| java.math | see CDC |
| java.net | see CDC |
| java.security | see CDC |
| java.security.cert | see CDC |
| java.security.acl | access control lists |
| java.security.interfaces | interfaces for generating keys |
| java.security.spec | key specifications, and algorithm parameter specifications |
| java.text | see CDC |
| java.text.resources | see CDC |
| java.util | see CDC |
| java.util.jar | see CDC |
| java.util.zip | see CDC |
| javax.microedition.io | see CDC |

The profile products page is here:
<http://java.sun.com/products/foundation/>

J2ME RMI PROFILE (JSR #66)

This profile interoperates with J2SE RMI, and provides Java platform-to-Java platform remote method invocation for Java devices.

J2ME GAME PROFILE (JSR #134)

This is a proposed Micro Edition specification, so nothing is yet defined. According to the JCP home page for JSR #134 (the Game Profile), the following areas will be covered:

- 3D Modeling and Rendering for Games
- 3D Physics Modeling for Games
- 3D Character Animation for Games
- 2D Rendering and Video Buffer Flipping for Games
- Game Marshalling and Networked Communication
- Streaming Media for Games
- Sound for Games
- Game Controllers
- Hardware Access for Games

PDA PROFILE (JSR #75)

The PDA Profile will provide UI and storage APIs for small, resource-limited handheld devices.

PERSONALJAVA SPECIFICATION – VERSION 1.2A

| | |
|--|--|
| java.applet | full support from JDK1.1.8 |
| java.awt | modified from JDK1.1.8 |
| – note: there is an extra method for PJ for double-buffering in java.awt.Component | |
| java.awt.datatransfer | full support |
| java.awt.event | full support |
| java.awt.image | full support |
| java.awt.peer | modified |
| java.beans | full support |
| java.io | modified |
| java.lang | modified |
| java.lang.reflect | modified |
| java.math | optional – may or may not be supported |
| java.net | modified |
| java.rmi | optional |
| java.rmi.dgc | optional |
| java.rmi.registry | optional |
| java.rmi.server | optional |
| java.security | modified |
| java.security.acl | unsupported |
| java.security.cert | some classes required, some optional |
| java.security.interfaces | required if code signing is included |
| java.security.spec | required if code signing is included |
| java.sql | optional |
| java.text | full support |
| java.text.resources | modified |
| java.util | modified |
| java.util.jar | required if code signing is included |
| java.util.zip | modified |

Additional PersonalJava specific packages are:

| | |
|--------------|---------------------------------------|
| com.sun.awt | for mouseless environments |
| com.sun.lang | a couple of error & exception classes |
| com.sun.util | for handling timer events |

PersonalJava will eventually be superseded by the Personal Profile. For more information on the PersonalJava Application Environment:
<http://java.sun.com/products/personaljava/>

JAVA TV – VERSION 1.0

| | |
|-----------------------------|--|
| javax.tv.carousel | access to broadcast file and directory data |
| javax.tv.graphics | root container access and alpha blending |
| javax.tv.locator | referencing data and resources |
| javax.tv.media | controls and events for management of real-time media |
| javax.tv.media.protocol | access to generic streaming data in a broadcast |
| javax.tv.net | IP datagram access |
| javax.tv.service | service information access |
| javax.tv.service.guide | supporting electronic program guides |
| javax.tv.service.navigation | services and hierarchical service information navigation |
| javax.tv.service.selection | select a service for presentation |
| javax.tv.service.transport | information about transport mechanisms |
| javax.tv.util | creating and managing timer events |
| javax.tv.xlet | communications interfaces used by apps and the app manager |

Get off that couch and check out the JavaTV page at the following URL:
<http://java.sun.com/products/javatv/>

JAVA EMBEDDED SERVER – VERSION 2.0

| | |
|-------------------------------------|---|
| com.sun.jes.service.http | servlet/resource registrations |
| com.sun.jes.service.http.auth.basic | http basic authentication |
| com.sun.jes.service.http.auth.users | management of users and their access |
| com.sun.jes.service.timer | for handling timer events |
| org.osgi.framework | consistent model for app. dev., supports dev. and use of services |
| org.osgi.service.device | detection of devices |
| org.osgi.service.http | http access of resources |
| org.osgi.service.log | logging facility |

You can find more information on Embedded Server on the following site:
<http://www.sun.com/software/embeddedserver/>

JAVA CARD – VERSION 2.1.1

| | |
|--------------------|--|
| java.lang | fundamental classes |
| javacard.framework | core functionality of a JC Applet |
| javacard.security | security framework |
| javacardx.crypto | extension package with security classes and interfaces |

Next time you use that American Express Blue card, you may want to know how it works, so take a look here:
<http://java.sun.com/products/javacard/>

sitraka

www.sitraka.com

J2ME A REVIEW BENCHMARKING

Evaluating the performance of a JVM objectively Written by
Carl Barratt
& Glenn Coates

It could be argued that the clock speed of a given processing platform enables you to estimate the execution time of a user application running on that platform.

However, quoting figures such as MIPS (millions of instructions per second) are somewhat futile, since the execution of a specific number of instructions on one processor will not necessarily accomplish the same end result as that same number of instructions running on a different processor. It's the execution speed of a given set of instructions that's of greater concern when selecting an appropriate platform to run application code.

Clearly some platforms will be more proficient than others in this regard, though this is a difficult parameter to quantify since it's dependent to a large extent upon the application code in question. Benchmarking is the technique used to measure the speed at which a particular platform is able to execute code. Indeed, this is evident in the abundance of benchmarks available. Numerous examples of Java benchmarking are listed at www.epcc.ed.ac.uk/javagrande/links.html.

Benchmarks vary significantly in their complexity, but invariably they comprise a number of lines of code that, when executed on the platform being tested, generates a discrete value to use during its appraisal. This facilitates a comparison of the execution speed with similar platforms. Typically there are three types of benchmarks, which have inherited titles in accordance with their origin:

- User
- Manufacturer
- Industry

User benchmarks are, as the name suggests, created by any individual with an interest in the field. Countless examples are available and characteristically they vary in quality; in the past benchmarks of this type have been very influential.

Market incentives have driven the introduction of manufacturer benchmarks; invariably these are written to benefit the platform in question and so can be disregarded unless used to facilitate the relative performance of platforms offered by that particular vendor.

Finally, the financial significance of benchmarking has resulted in the development of industry benchmarks, which are usually considered to be of high integrity. Such benchmarks are defined by an independent organization, typically composed of a panel of industry specialists.

Why Write a Paper on Java Benchmarking?

Results are published for multiple benchmarks and the primary issues can be clouded by hype; as a consequence the selections available to the end user are somewhat overwhelming. The crucial point is how well your code performs on the chosen system, so the question is: How do you identify a benchmark that best models your application? An understanding of benchmarks is vital to enable the user to select an accurate measurement tool for the platform in question and not be misled by the results.

The purpose of this article is to educate device manufacturers, OEMs, and, more specifically, J2ME development engineers, while at the same time resolving any remaining anomalies in a discipline that's commonly misunderstood.

What Is a Benchmark?

Fundamentally, a benchmark should incorporate programs that, when invoked methodically, exhaustively exercise the platform being tested. Implicit in this process is the generation of a runtime figure corresponding to the execution speed of the platform. Benchmarks can be simplistic, comprising a sequence of simple routines executed successively to check the platform's response to standard functions (e.g., method invocation). Typically, both the overall elapsed time and that for each routine in isolation is considered; in the former case it's usual to assert a weighting coefficient to each routine that's indicative of its relevance in the more expansive context. Each routine should run for a reasonable amount of time. The issue here is an assurance that performance statistics are not lost within overheads at start-up.

pramati

www.pramati.com



Benchmarks can also be more substantive; for example, processor-intensive applications can check multithreading by running several other routines simultaneously to evaluate context switching. Essentially there's no substitute for running the user's own application code on the platform in question. However, while this argument is laudable, it's beyond reasonable expectation that the platform manufacturer can implement this. To facilitate an accurate appraisal, it's vital that any standard benchmark utilized by competing manufacturers

Java-Specific Benchmarks

As with other platforms, numerous Java benchmarks have appeared (see Table 1).

CaffeineMark is a pertinent instance of a benchmark since its results are among those most frequently cited by the Java community. On this basis we chose it as an example for further discussion.

CaffeineMark encompasses a series of nine tests of similar length designed to measure disparate aspects of a Java

“A benchmark should incorporate programs that, when invoked methodically, exhaustively exercise the platform being tested”

should mimic as much as possible the way the platform will ultimately be used.

The Advantages and Limitations of Benchmarking

Industry benchmarks are useful for providing a general insight into the performance of a machine. Still, it's important not to rely on these benchmarks since such a preoccupation distracts from the bigger picture. While they can be employed generally to realize the efficient comparison of different platforms, they have shortcomings when applied specifically. For example, one function may be heavily used in the application code when compared to another, or certain functions may run concurrently on a regular basis. There are inherent benefits in developing your own benchmark as this facilitates the tailoring of routines to imitate the end application or to expose specific inadequacies in peripheral support. Manufacturers' benchmarks can be written to aid the cause of specific vendors and so can easily be tailored to mislead.

When considering more restrictive embedded environments, such as those used by J2ME-compliant devices, it becomes apparent that the application developer must consider the risks inherent in the hardware implementation of a virtual machine prior to making a purchasing decision.

Speed is a primary consideration when adopting a JVM within restricted environments; implementations of the J2ME vary significantly in this respect, from JVMs that employ software interpretation and JIT compilers that compile the bytecode to target machine code while the application is being executed, to native Java processors offering much greater performance.

Other factors to consider include the response time of the user interface, implementation of the garbage collector, and memory issues since consumer devices don't have access to the abundant resources available to desktop machines. While this may seem a tangential point as far as benchmarking is concerned, it's one worth making since it's imperative that these areas in particular are comprehensively exercised. Subject to these caveats, benchmarking is a valuable technique that aids in the evaluation of processing platforms, and, more specifically, J2ME platforms.

Virtual Machine's performance. The product of these scores is then used to generate an overall CaffeineMark. The tests are:

- **Loop:** Employs a sort routine and sequence generation to quantify the compiler optimization of loops
- **Sieve:** Utilizes the classic sieve of Eratosthenes to extract prime numbers from a sequence
- **Logic:** Establishes the speed at which decision-making instructions are executed
- **Method:** Executes recursive function calls
- **Float:** Simulates a 3D rotation of objects around a point
- **String:** Executes various string-based operations
- **Graphics:** Draws random rectangles and lines
- **Image:** Draws a sequence of three graphics repeatedly
- **Dialog:** Writes a set of values into labels and boxes on a form

An embedded version of CaffeineMark is available that excludes the scores of the Graphics, Image, and Dialog tests from the overall score. Furthermore, CLDC doesn't support floating-point operations, so the "Float" test is ineffective in this context. This benchmark is regularly updated to account for vendor optimizations and continues to be a reasonably accurate predictor of performance for JVMs.

Bearing this in mind, alongside the high take-up of CaffeineMark in the industry, it's unfortunate that it's unsuitable for embedded environments such as J2ME. The cogency of this argument is based upon its inability to benchmark the interaction of Java subsystems, and the subsequent failure to imitate typical real-world applications faced by such devices. More specifically, it doesn't take into account certain situations in which a platform may have to cope with a heavily used heap, the garbage collector running all the time, multiple threading, or intensive user interface activities.

To address some of these issues, representatives of leading companies in the field have recently formed a committee under the banner of the Embedded Microprocessor Benchmark Consortium (EEMBC) to discuss the introduction of an industry benchmark for J2ME devices.

| BENCHMARK | URL | COMMENTS |
|-----------------------|--|---|
| SPEC JVM98 | www.spec.org/osg/vm98 | Suited to Java clients running JDK1.1 or higher |
| CaffeineMark 3.0 | www.webfayre.com/pendragon/cm3/index.html | Executes a series of standard routines to exercise JVMs |
| Linpak Java benchmark | http://netlib.org/benchmark/linpakjava | Utilizes a dense 500x500 system of linear equations |
| SciMark 2.0 benchmark | http://math.nist.gov/scimark2 | Aimed at end applications intensive in scientific and numerical computing |
| VolanoMark | www.volano.com/benchmarks.html | Primarily suited to evaluating the performance of Java servers |

TABLE 1 Examples of Java-specific benchmarks currently in existence

compuware

www.compuware.com

What Is EEMBC?

EEMBC (www.eembc.org) is an independent industry benchmarking consortium that develops and certifies real-world benchmarks for embedded microprocessors; the consortium is established among manufacturers as a yardstick for benchmarking in this context. A principal concern of the committee is to produce dependable metrics, enabling system designers to evaluate the performance of competing devices and consequently select the most appropriate embedded processor for their needs. The industry-wide nature of such committees intrinsically helps to combat the practice among some vendors of striving to artificially improve their ratings via special optimizations of the compiler, which is now so wretchedly prevalent.

“When applied prudently, benchmarks are an invaluable asset that aid in the selection of hardware to suit a particular application”

A subcommittee was recently formed under the umbrella of this organization to develop similar benchmarks for hardware-based virtual machines. Founding companies within the consortium include Vulcan Machines Ltd, ARM, Infineon, and TriMedia. Primarily the committee aims to identify the limitations of existing Java benchmarks, and to develop new ones in which “real-world” applications are afforded a higher priority than low-level functions.

An example benchmark conceived on this basis could be a Web browser. Since this is a very intensive end application in almost every respect, a figure relating to the proficiency of the device running low-level code in isolation wouldn't prove particularly representative of its functionality.

Consequently, the EEMBC consortium solution is expected to employ a series of applications reflecting typical real-world scenarios in which CDC- and CLDC-compliant devices can be employed. Further examples of such benchmarks include a generic game or organizer that exercises intensive garbage collection, scheduling, high memory usage, user interface, and dynamic class loading. This way system designers are able to evaluate potential devices for inclusion in their end application by the appraisal of a benchmark derived in an environment that's analogous to that application.

Other Considerations?

When applied prudently, benchmarks are an invaluable asset that aid in the selection of hardware to suit a particular application. However, they shouldn't be regarded as the sole criteria. It's imperative that J2ME-embedded system designers don't rely upon the use of benchmarks exclusively, since the issue is clouded by many other factors.

In the context of J2ME, systems extend beyond the virtual machine to its interaction with peripheral devices such as a memory interface; clearly such peripherals and the interfaces to them must be considered when measuring the time it takes to execute an application. In the case of memory, limitations will be imposed on a J2ME-optimized device; this raises numerous issues that may impact the performance of the device, for example, garbage collection.

Also, implicitly, batteries are employed to power hardware that's compliant with the CLDC specification. Consequently, power consumption of the virtual machine is of primary con-

cern and, accordingly, the clock speed must be kept to a minimum. For example, it's pertinent here that while software accelerators may post acceptable benchmark scores, they may also, as a consequence of their reliance upon a host processor, consume excessive power compared to a processor that executes Java as its native language.

Another significant factor is the device upon which the virtual machine is implemented. The FPGA or ASIC process used will clearly affect the speed at which the processor runs, and variations in benchmark scores are a natural corollary of this. Furthermore, the silicon cost of the entire solution that's required to execute Java bytecode must be considered, particularly where embedded System-on-Chip implementations of the JVM are concerned. Similarly, the

designer should be aware of fundamental issues such as the “quality” of the JVM in terms of compliance with the J2ME specification, reliability, licensing costs, and the reputation of the hardware vendor for technical support. All these factors must be considered in tandem with the benchmark score of the virtual machine prior to making a purchasing decision.

Conclusion

No benchmark can replace the actual user application. At the earliest possible stage in the design process, application developers must run their own code on the proposed hardware, since similar applications may post a significant disparity in terms of performance on the same implementation of the virtual machine. However, since designers are often focused on using their time more productively, they frequently rely upon industry benchmarks for such data. While there's no panacea, industry benchmarks such as that proposed by EEMBC are a useful tool to aid in the evaluation of performance, provided you're aware of its limitations in a J2ME environment. ●

Resources

- Coates, G. “Java Thick Clients with J2ME.” *Java Developer's Journal*. Vol. 6, issue 6.
- Coates, G. “JVMs for Embedded Environments.” *Java Developer's Journal*. Vol. 6, issue 9.
- Cataldo, A. (April, 2001). “Java Accelerator Vendors Mull Improved Benchmark.” *Electronic Engineering Times*.

AUTHOR BIOS

Glenn Coates works for Vulcan Machines as a VM architect developing a Java native processor called Moon and has been a software engineer for nine years. For the last four years he has worked with mobile devices and Java developing products. He also represents his company at the EEMBC meetings. Glenn holds a degree in computer science and is also a Sun-certified architect for Java technologies.

Carl Barratt works in applications support for Vulcan Machines. He has over seven years of experience in various hardware and software development roles. Carl holds a BEng (Hons) degree in electronic engineering and has undertaken PhD research at the University of Nottingham.

▼▼▼ glenn@vulcanmachines.com & carl@vulcanmachines.com

softwired

www.softwired.com

Hardware Accelerators for J2ME Come of Age

WRITTEN BY
RON STEIN



Embedded Java technology, specifically the J2ME platform, provides a universal and secure runtime platform for transient, application-based content and services. Through the “write once, run anywhere” promise of Java, the J2ME platform can support a range of Internet appliances, from mobile wireless devices to TV set-top boxes.

Companies that deploy Java-enabled devices with native and resident applications developed using Java benefit because porting isn't necessary when the underlying device technology changes. These benefits have caused companies – such as J-Phone, Motorola, Nokia, NTT DoCoMo, and Sprint – to integrate Java technology into their products and services.

The flexibility afforded by the Java platform has typically meant slow and, at times, inadequate Java software execution and high-system energy usage that impacts battery life. Despite today's deployment of Java-enabled devices, the performance issue remains.

As with Java applications running on desktop computers and servers, the goal of Java-enabled devices is to achieve Java software execution and energy use that is equivalent to compiled software developed in C, C++, and assembly code. The acceleration techniques used for desktop computers and servers,

however, are inappropriate for Internet appliances and wireless mobile devices that are typically cost-sensitive and powered by batteries.

Computing- and processing-intensive algorithms commonly migrate from initially being implemented in software to being in dedicated hardware. For example, generating graphics images, such as drawing polygons, moving sprites, and rendering 3D effects, were all initially implemented as software routines. Now these types of graphics functions are handled by the graphics processor chip, where the specialized processing can be done faster and more efficiently than by software running on the main system microprocessor. A similar migration is under way with regard to executing Java bytecode instructions, hence boosting the performance of Java application software developed for the J2ME platform.

One of the big benefits of processing Java bytecode instructions in dedicated

hardware is to bridge the gap between the stack-based Java runtime and register-based microprocessors. The issue here is that the Java stack operations require memory transactions to interact with the stack (typically held within memory), which slows performance and burns energy.

Recently, In-Stat/MDR-released findings point to more than 50% of embedded JVMs utilizing hardware acceleration by 2004, increasing to better than 70% in 2005. Device designers and manufacturers, who are today designing products that include hardware solutions to boost Java software execution, validate the attraction of hardware acceleration. These market forces, combined with the natural trend to migrate these computationally intensive activities to hardware, have enabled hardware accelerators for the J2ME platform to come of age.

Before discussing Java acceleration hardware further, it's important to understand that hardware Java accelerators don't replace a Java Virtual Machine (JVM) but rather complement it.

A Java Virtual Machine has several components, only one of which (the Java bytecode instruction interpreter loop) is enhanced or replaced by a hardware-based acceleration technology. The JVM still takes care of loading and verifying Java classes, managing memory, scheduling tasks, and other house-keeping functions. The more efficiently these other functions are implemented, the greater the effect an accelerator can have on overall performance.

Java gets a needed boost

//

The acceleration technology most likely to be adopted and included into devices must complement existing designs and not force major changes

//

vmgear

www.vmgear.com

Acceleration Technologies

All acceleration technologies primarily focus on the main bottleneck within the JVM – the interpretation process – and seek to improve the speed at which Java bytecode instructions are interpreted and/or executed. A second area of focus for some accelerators is to help speed up the garbage collection process.

Supporting the market's appetite for Java acceleration hardware are a number of solutions that can be grouped into three main categories: native Java microprocessors, Java accelerator peripherals/coprocessors, and integrated Java bytecode interpreters.

Native Java microprocessors use Java bytecode instructions as the native instruction set. It should be noted that these microprocessors might also include additional instructions necessary to support an operating system and device drivers.

Java accelerator peripherals/coprocessors are standalone chips or components within a System on a Chip (SOC) that execute Java bytecode instructions.

Integrated Java bytecode interpreters, including instruction set extensions that support bytecode execution and other JVM operations, are micro-

processor architecture extensions that perform on-the-fly Java bytecode interpretation within the core of a microprocessor chip.

According to In-Stat/MDR, native Java microprocessors “are an anti-infrastructure solution” because development tools, add-ins, and other support are not available from the third-party market at large. The absence of infrastructure and third-party market support makes the overall solution unpalatable and is one of the primary reasons that Sun Microsystems abandoned the development of its own native Java microprocessors (the “Pico Java” and “Micro Java” microprocessors).

Native Java microprocessors in general continue to be poorly received by the marketplace.

Java accelerator peripherals/coprocessors are the newest category of devices and hold great promise as long as they can be easily and transparently integrated into a system. As with native Java microprocessors, the more effort that it takes for a designer to incorporate a media or content accelerator, the less likely that technology or product will be chosen.

What this means is that the acceleration technology most likely to be adopt-

ed and included into devices must complement existing designs and not force major changes. In today's market, designers prefer to have a hardware Java accelerator and JVM as an overall integrated solution. A solution that recognizes and accommodates designers' requirements will be embraced more rapidly than more intrusive options.

A key attribute and benefit of any solution is one that offers transparent integration with any JVM as well as with existing hardware designs that minimize development efforts and therefore reduce the always-critical time-to-market.

Choosing the Right Solution

As the J2ME platform gains acceptance in wireless mobile devices, Internet appliances, and other embedded applications, that acceptance is, in turn, driving adoption of specialized hardware that improves the performance of software developed for the J2ME platform. Different options exist in today's market, each with its own unique set of performance characteristics and pros and cons, so designers are encouraged to assess their requirements in advance and choose a solution accordingly. ☛

▼▼▼ ron@nazomi.com

AUTHOR BIO

Ron Stein is a senior marketing manager at Nazomi Communications, Inc. He has an MBA from Santa Clara University and a BSEE from the University of Pittsburgh.

corda

www.corda.com

infragistic

www.infragistics.com



Scandalous Propaganda:
‘Twenty-Eight Times Faster than J2EE!’

—continued from page 5

site, but it was actually fully owned/operated by Microsoft (www.gotdotnet.com/team/compare/). It's a bit of a shame, as that instantly puts us Java developers on the defensive when browsing it, especially when you read the opening gambit that they claim to have implemented the J2EE Pet Store application using .NET technology and made it run 28 times faster at only a fraction of the cost. Mmmmm...anyone that believes that on face value is heading for trouble.

Being a little cynical I had to find out more. I spent a lot of time looking through their documentation, their examples, their white papers, and most of all their figures. Irrespective of their conclusions, what annoys me is the fact that it's so obviously weighted in Microsoft's favor. Had this been a third-party report, then maybe that would have lent a little more credence to it, but, as it is, you have to take it with a pinch of salt.

IBM wasn't too impressed with the findings either, it would appear. They issued a counterreport detailing all the flaws in Microsoft's claim. Microsoft had chosen to demonstrate their findings on a number of

different configurations, including Oracle, Sun, and IBM. Although, as far as I'm aware, IBM is the only one to publicly denounce the findings. Maybe Sun and Oracle are just too busy for a retort!

When you read the IBM paper, "Setting the Record Straight" (www-3.ibm.com/software/info1/websphere/news/ibmnews/compview4.jsp), it does indeed prove interesting reading and on the whole offers a well-balanced argument to the Microsoft claim that J2EE is 28 times slower than .NET. However, that's not to say IBM's report is perfect – they make some leaps-of-faith as well, although not as glaring as what Microsoft is leading you to believe. I advise you to access the two reports, have a read, and draw your own conclusions.

This was debated hotly on our mailing list (Straight Talking@Yahoo), but sadly the majority of the posters are hard-core Java developers, so we didn't have anyone with .NET experience to argue against. This, I think, is the crux of the majority of debates: .NET developers are seriously thin on the ground. To that end, Microsoft is very good at convincing nondevelopers that they are developers and, therefore, I don't think we'll have a good debate for a long time yet. Microsoft is good at empowering their community; develop an MS Word Macro and you'll find yourself being called a "Web services" devel-

oper! Since the word "developer" has a sexier ring than most titles, a lot of people are happy to have this title bestowed on them and begin to believe they are real developers. Dangerous state of affairs.

This whole debate between .NET and J2EE could well be academic. At the end of day, as one poster quite correctly pointed out, we didn't consciously choose TCP/IP or HTTP. We chose them because everyone else did. It was the beehive mentality with respect to technology standards; we just swarmed to the most popular standard.

I believe the decision between .NET and J2EE will probably come down to the largest user body. If that happens to be .NET, then we'll start seeing a lot more crossover from the J2EE vendors to ensure they don't lose their customer base. If it's J2EE, then Microsoft will have to concede and start moving "C#" back to Java again. (How does that saying go? "It will be a cold day in hell before that happens.")

I look forward to what 2002 will bring to our Java lives. But be warned, it's not going to be an easy year for us Java developers. Our loyalties are going to be tested. We're going to be lured to the "dark side" with tales of vast riches and connected, interoperable systems. We have to resist.

We can't be assimilated to the Microsoft Borg. If we are, then the computing industry will be a desperately bland and boring place to work. ☘

basebeans

www.basebeans.com

javaone
www.sun.com

SAVE 30% Off*
the annual
newsstand rate

JAVA DEVELOPER'S JOURNAL

* Offer subject to change without notice

| ANNUAL NEWSSTAND RATE | |
|-----------------------|------------------------|
| \$71.88 | |
| YOU PAY | |
| \$49.99 | |
| YOU SAVE | |
| 30% | Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of *Java Developer's Journal* for only \$49.99! That's a savings of \$21.89 off the cover price. Sign up online at www.sys-con.com or call 1-800-513-7111 and subscribe today!

Here's what you'll find in every issue of *JDJ*:

- Exclusive coverage
 -  J2EE
 -  J2SE
 -  J2ME
- Exclusive feature articles
- Interviews with the hottest names in Java
- Latest Java product reviews
- Industry watch



Prescriptions for Your Java Ailments



Unlike the doctor who works for your HMO, I won't require a copayment for each visit nor ask you to fill out long arduous forms. I'm here to help readers of *Java Developer's Journal* find a cure for their Java system ills.



HOW CAN I ACCESS THE ACTIVE DIRECTORY OF WINDOWS 2000 SERVER TO AUTHENTICATE REGISTERED USERS WITH MY APPLICATION?

I have a couple of thoughts. First, I commend you on being open-minded about using the Active Directory to store user information for your Java-based applications. Many of us are anti-Microsoft, and it gets in the way of deploying systems rapidly and in a cost-effective manner. While Microsoft did deviate from the standard LDAP implementation, it will more than meet your needs for the future. I like the fact that it frees you from being stuck with per-entry charges as in other implementations.

Now for your answer. You can authenticate to the Active Directory using simple authentication by setting the following JNDI security properties:

```
javax.naming.security.authentication
javax.naming.security.principal
javax.naming.security.credentials
```

Let's look at a code snippet to demonstrate how to authenticate the administrator in the domain doctorjava.com:

```
// Create a table to store the five
// properties
Hashtable env = new Hashtable(5);

env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtx
Factory");
env.put(Context.PROVIDER_URL,
```

```
"ldap://pdc.doctorjava.com:389");
env.put(Context.SECURITY_AUTHENTICATION,
"simple");
env.put(Context.SECURITY_PRINCIPAL,
"cn=admin, cn=users, dc=pdc, dc=doc-
torjava, dc=com");
env.put(Context.SECURITY_CREDENTIALS,
"doctorjava");
try {
    ctx = new InitialDirContext(env);
    // Do work
}
```

The provider URL should point to a domain controller. It would be ideal to also change the port that the Active Directory LDAP provider listens on. Instructions on how to do this are located at <http://msdn.microsoft.com>.



I WOULD LIKE TO USE A JAVA.NET.URL OBJECT, BUT I'M BEHIND A PROXY SERVER. WHAT DO I NEED TO DO TO GET IT TO WORK?

There are two different approaches to solving your problem. Essentially, you can specify the Proxy server as a system property when you start up the JVM or do it programmatically. If you want to do it while starting the JVM, here's the syntax:

```
Java -classpath -DproxySet=true
-DproxyHost=myproxy -DproxyPort=80
```



ctia wireless

www.ctia.com

SUBSCRIBE AND SAVE

XML JOURNAL

Offer subject to change without notice

| ANNUAL NEWSSTAND RATE | |
|-----------------------|------------------------|
| \$83.88 | |
| YOU PAY | |
| \$77.99 | |
| YOU SAVE | |
| \$5.89 | Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of *XML-Journal* for only \$77.99! That's a savings of \$5.89 off the annual newsstand rate.

Sign up online at www.sys-con.com or call 1-800-513-7111 and subscribe today!

In January XML-J:

Got XSLT? – Part 3

Transform an XML example to speech

XSL Formatting Objects – Here Today, Huge Tomorrow

XSL-FO, while in its early stages, is poised to become the 'next big thing' to present XML data

Using the IBM XML Security Suite – Part 2

Encrypting XML documents dynamically

SOAP Messages with Attachments

How the emerging W3C note can be used with the Apache SOAP implementation



You can also set it programmatically in your code by setting the system properties as follows:

```
System.getProperties().put("proxySet", "true");
System.getProperties().put("proxyHost", "my-proxy");
System.getProperties().put("proxyPort", "80");
```

I WAS READING THAT WINDOWS XP NOW PROVIDES RAW SOCKETS SUPPORT. I'M INTERESTED IN USING JAVA TO DEVELOP A NETWORK SNIFFER AS WELL AS AN EQUIVALENT TO TRIANGLEBOY (WWW.TRIANGLEBOY.COM). COULD YOU POINT ME IN THE RIGHT DIRECTION?

I see the potential for mischief in answering this question but will leave it up to each reader's individual judgment. First, Java doesn't natively provide access to raw sockets. I see the merit in someone taking the lead in developing a specification for raw sockets support as part of the Java Community Process (www.jcp.org). Luckily, there is a third-party API (JPCAP) that allows you to capture and send IP packets from a Java application. For more information, visit www.goto.info.waseda.ac.jp/~fujii/jpcap/index.html.

The JPAP APIs currently support Ethernet, Ipv4, Ipv6, ARP/RARP, TCP, UDP and ICMPv4. It has been tested on FreeBSD 3.x, Linux RedHat 6.1, Solaris, and Microsoft Windows 2000.

I WAS READING THE JAVAMAIL SPECIFICATION ON THE SUN SITE AND SAW PROVIDERS FOR POP3 AND IMAP. I'D LIKE TO KNOW IF THERE'S A JAVAMAIL PROVIDER THAT SUPPORTS NNTP?

You're in luck. Knife will do everything you require from a news-user agent. Visit <http://dog.net.uk/knife/>.

I'M LOOKING FOR A GOOD UML MODELING TOOL. RATIONAL ROSE AND TOGETHER/J ARE OUT OF MY BUDGET. COULD YOU POINT TO ANY GOOD BUT FREE TOOLS THAT PROVIDE SIMILAR FUNCTIONALITY?

Many developers' tools are pricey. Even large corporations can't afford to put tools on all their developers' desks and have to either use these tools illegally or deploy some floating license scheme that causes its own problems. The open source movement seems to be gath-

ering speed and is targeting users like you with cost-effective (if not free) answers to your problems.

One tool you should consider looking at is ArgoUML (<http://argouml.tigris.org>). It supports the latest UML specification, is completely open source, and written in Java. It's actually easy to use and provides guidance to modelers when they're making design decisions.

While you're on its site, check out Subversion (<http://subversion.tigris.org>), which is a good version control system. Also look at Scarab (<http://scarab.tigris.org>), which is a Web-based issue tracking system.

DO YOU KNOW OF A GOOD JAVA OBFUSCATOR?

Many of my peers are familiar with my "Eschew Obfuscation" screen saver (my way of saying don't make simple things hard to understand). When I am in Dilbert mode, I sometimes think some developers are prone to obfuscation. Just look at some of the code out there.

The main reason you'll want to use an obfuscator for your programs is to protect some portion of your code from being reverse-engineered. These programs work by renaming human-understandable variables and method names to something that is less obvious. Usually you'll use this to protect license and registration information. There are many obfuscators on the market (such as RetroGuard, Source Guard, DashO-Pro, and JODE) that will address your needs. None of them can prevent reverse-engineering 100%, but they will keep junior developers from understanding the code.

HOW CAN I PRINT AN HTML/XML DOCUMENT FROM A JAVA PROGRAM USING J2EE CLASSES TO POSTSCRIPT?

I'd like to thank Renu Verma for coming up with a question that stumped me for a couple of days. But persistence prevailed and I have the answer. Java currently does not have sup-



sys-con media

www.sys-con.com

SAVE 30% Off*
the annual newsstand rate

BUSINESS & TECHNOLOGY
wireless

* Offer subject to change without notice

| |
|-----------------------------------|
| ANNUAL NEWSSTAND RATE |
| \$71.88 |
| YOU PAY |
| \$49.99 |
| YOU SAVE |
| 30% Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of **Wireless Business & Technology** for only \$49.99! That's a savings of 30% off the cover price. Sign up online at www.sys-con.com or call 1-800-513-7111 and subscribe today!

Wireless Business & Technology:

i-mode 101

A *WBT* Special for all wireless operators and developers around the globe hoping to quickly learn how to imitate NTT DoCoMo's wildly successful i-mode data offering in Japan.

Is Wireless Broadband Barreling Your Way?

WBT's David Geer looks at wireless broadband connectivity as it spreads around the globe.

The Impact of Privacy on the Future of Wireless Advertising

WBT's SMS editor, Dan Lubar, questions the balance of location-based advertising and an individual's right to privacy.

The GSM Association M-Services Initiative

A look at a series of guidelines generated to give WAP a friendlier consumer face.



port for image-saving functionality, but this will be introduced as part of JDK 1.4 where it will add a new image I/O package. The functionality is based upon JSR 15 – Image I/O framework (<http://java.sun.com/aboutJava/communityprocess/review/jsr015/index.html>).

In the meantime, there are a couple of third-party add-ons you could use to get the functionality you desire. One product that works pretty well is provided by Lightsaber Computing and can be downloaded at www.lightsaber.com/project3PO/java2d-pstext.html. You should also visit Snowbound's Software RasterMaster (www.snowbnd.com). On the chance that you also need to read a postscript document, check out ToastScript (www.geocities.com/toast-script/).

WHAT JDBC DRIVERS ARE AVAILABLE FOR MICROSOFT SQL SERVER?

You're in luck. If you're using SQL Server 2000, Microsoft currently has JDBC drivers in beta. You can check out www.jturbo.com/ for drivers. If you're on an earlier platform, you'll need to consider a third party. The highest quality drivers I've run across to date are from BEA, but they're pricey. If you're looking for something cheap (or even free) I recommend visiting www.freetds.org. Their drivers work pretty well.

I'M USING SQL SERVER AND WOULD LIKE TO MAKE IT BEHAVE LIKE JAVA.UUTIL.STACK SO THAT MULTIPLE USERS CAN DO POP() BUT NOT RETRIEVE THE SAME ROWS.

It took me a long time to come up with the right prescription (actually two different ones). The first solution involves putting logic in the database (faster but not portable), while the second shows how to do it in Java. Let's describe the database approach in detail. For example, you have a table that looks like the following:

| NAME | TYPE |
|-------------|---------------|
| UID | Int |
| Name | Varchar(200) |
| DateUpdated | Smalldatetime |

You'll need to add another column to the table so you can provide a simulated lock mechanism. We'll wrap this column in a transaction so that it will handle concurrency issues. You'll also need to retrieve only the first value. The following code ties together all of the above requirements:

```
DECLARE @UID INT
BEGIN TRAN
SELECT TOP 1 @UID = UID FROM
RegisteredUser WHERE LOCKSTATUS IS
```

NULL

```
UPDATE RegisteredUser
SET LOCKSTATUS = 1
WHERE UID = @UID
```

```
SELECT * FROM RegisteredUser WHERE UID = @UID
```

COMMIT TRAN

As an alternative, you could move the logic into Java and control the transaction behavior yourself. I hope this meets your needs.

I KNOW HOW TO START A NEW PROCESS, BUT NEED A WAY TO TELL IF A PROCESS IS ALREADY RUNNING. THE SYSTEM AND RUNTIME CLASSES DON'T SEEM TO PROVIDE A MECHANISM.

You're correct that System and Runtime classes don't provide any mechanism for determining already running processes. Most likely this is for reasons of security and platform independence. Luckily there's a third-party way of getting at what you desire, see www.jconfig.com.

I'M NEW TO JAVA AND AM LOOKING FOR AN EQUIVALENT TO C++ PRINTF SO THAT I CAN WRITE INTEGERS, FLOATS, AND OTHER DATA TYPES TO DISK.

The DataOutputStream class provides all the functionality you desire. It provides methods to write primitive data types to any output stream (network connections, files, and so on). You can use the DataInputStream class to read data back in.

I'D LIKE TO USE JMS ALONG WITH IP MULTICAST. COULD YOU POINT ME IN THE RIGHT DIRECTION?

There are several JMS providers that support multicast messaging but my favorite is



sys-con media

www.sys-con.com

SUBSCRIBE AND SAVE

WebServices JOURNAL

Offer subject to change without notice

| ANNUAL NEWSSTAND RATE | |
|-----------------------|------------------------|
| \$83.88 | |
| YOU PAY | |
| \$69.99 | |
| YOU SAVE | |
| \$13.89 | Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of **Web Services Journal** for only \$69.99! That's a savings of \$13.89 off the annual newsstand rate. Sign up online at www.sys-con.com or call 1-800-513-7111 and subscribe today!

In January **WSJ**:

Invoking .NET Web Services from Mobile Devices
Pocket PCs put .NET Web services at your fingertips

It's More Than Just the Plumbing
The real issues are the nontechnical ones

Wireless Web Services with J2ME
Remote possibilities

Web Services @ Work
Gluing Web services to Baan

Bringing Web Services to Smart Devices
Realizing the value of the connected world

Web Services – Tiptoeing Through the Snarl
Creating Web services



FioranoMO. They provide many samples that are exactly what you're looking for. In addition, their implementation of using JMS with IP Multicast is a serverless model.



I'VE HEARD THAT IT'S A GOOD IDEA FOR A WEB SITE TO STORE A USER'S PASSWORD HASHED. COULD YOU TELL ME THE BEST WAY TO ACCOMPLISH THIS?

I can recommend a few. First, the password store will be more secure if the password uses a one-way algorithm. This, of course, will mean that you'll never be able to tell the user the password he or she entered. Second, I recommend that you immediately encrypt it in your application server instead of performing this function within the database before handing it off to your data tier. This way it can't be sniffed traveling over the wire. Third, I hope that you're using certificates on your Web site.

Now I'll show some simple code that uses the MD5 algorithm (one-way). To use this code, you'll need to download the Java cryptography extensions (JCE) from the Sun site.

```
import java.security.*;

public class Encrypt {

    private String password = null;

    public String new2Password( String passwd )
    {
        try {
            MessageDigest md=MessageDigest.get
            Instance("SHA-1");
            String clearPassword = passwd;
            md.update(clearPassword.getBytes());
            byte[] digestedPassword = md.digest();
            return new String(digestedPassword);
        }
        catch (java.security.NoSuchAlgorithm-
        Exception e) {
            System.out.println("Rats, MD5 doesn't
            exist");
            return null;
        }
    }

    public void setPassword(String passwd) {
        try {
            MessageDigest sha =
            MessageDigest.getInstance("MD5");
            byte[] tmp = passwd.getBytes();
            sha.update(tmp);
            password = new String(sha.digest());
        }
        catch (java.security.NoSuchAlgorithmException e) {
            System.out.println("Rats, MD5 doesn't exist");
        }
    }
}
```

```
}
}
}
```

As you can see the code is straightforward. You can use it as part of your registration and login routines. Simply insert it anywhere the user types his or her password and run it through this code before comparing it to the password stored in the database.



I'M PLANNING TO USE JAAS FOR SECURITY AND I SEE THAT THERE ARE TWO POLICY FILES, ONE IN JAVA.SECURITY AND THE OTHER IN JAVAX.SECURITY.AUTH PACKAGE. IF I USE THE JAVA.SECURITY FILE, DO I HAVE TO USE BOTH POLICY CLASSES? IF NOT, HOW CAN I TELL THE JVM TO USE EITHER ONE OF THE POLICY CLASSES.

Use of policies is a topic that can be confusing, but I'll point you in the right direction. Some of the confusion toward policies will be solved in a future JDK where there will be a single policy file. In the meantime, let's talk about the current state of affairs.

The default policy usually is an implementation of java.security.Policy. If you're using Java Authentication and Authorization Services (JAAS) and principal-based permission entries, you need to use an implementation of javax.security.auth.Policy. Depending on the particular JAAS provider, the actual policies may be stored locally in a file or externalized to a policy server (Netegrity uses this approach) on another tier.



IN A PREVIOUS ISSUE (JDJ, VOL. 6, ISSUE 11), YOU SHOWED HOW TO USE JAVA WITH A PROXY SERVER. THE TECHNIQUE YOU SHOWED DOESN'T WORK WITH OUR PROXY SINCE WE'RE SOCKS-BASED. COULD YOU TELL ME HOW TO GET THIS WORKING?



sys-con media

www.sys-con.com

I gave myself one demerit for forgetting about the other camp. Using a SOCKS-based proxy is just as simple. The following code snippet demonstrates how to set the appropriate properties:

```
Properties prop = System.getProperties();
prop.put("socksProxyPort", "1080");
prop.put("socksProxyHost", "socks.thehartford.com");
System.setProperties(prop);
```

Alternatively, you could specify the properties when you start the virtual machine:

```
java -DsocksProxyPort=1080 -DsocksProxyHost=socks.thehartford.com
```

If there are other types of proxy servers on the market that don't support either of the two recommended approaches, please don't hesitate to drop me a note.

I'M DEVELOPING AN RMI SERVER THAT MAKES SOME JNI CALLS TO DLLS. THE PROBLEM IS THAT WHEN I SHUT DOWN THE SERVER, I WANT IT TO CALL A CLEAN-UP FUNCTION IN THE DLL. I'VE ALREADY TRIED FINALIZERS.

You've discovered that finalize is not guaranteed to be called by the VM. Besides taking this approach would make the results of your code very unpredictable. Java JDK 1.3 and later provide the antidote to your poison. Take a look at addShutdownHook, which is part of the java.lang.Runtime package.

HOW CAN OR SHOULD I SUBCLASS HTTPURLConnection SO I CAN, FOR EXAMPLE, OVERRIDE SETREQUESTMETHOD() TO ALLOW THE SETTING OF CUSTOM METHODS?

HOW CAN I USE HTTPURLConnection AND PIPELINE MULTIPLE REQUESTS USING THE SAME CONNECTION?

I WOULD LIKE TO USE HTTPURLConnection TO RETRIEVE FILES FROM A REMOTE SERVER BUT AM RUNNING INTO ISSUES WITH LONG TIME-OUTS. IS THERE A WAY TO SPECIFY A TIME-OUT VALUE?

It's unusual to receive three questions regarding the same topic. I've decided to combine the answer to all three questions into one response since it should cover all three questions.

You've already figured out that there is no simple way of overriding the behavior of setRequestMethod as the compiler won't allow you to do so.

The second and third questions are similar in that they're all related to the default usage of sockets. HttpURLConnection doesn't allow us to pass in a socket connection for reuse nor does it allow us to specify a time-out value for the socket. While the RFCs allow you to reuse a socket, its actual implementation is up to the application server you're using. The time-out value of a socket defaults to the operating sys-

“The main reason you'll want to use an obfuscator for your programs is to protect some portion of your code from being reverse-engineered.”

RECEIVE \$150
DISCOUNT OFF FULL CONFERENCE
REGISTRATION

web services **EDGE**
conference & expo

2002 WORLD TOUR

Learn How to Develop
SOAP Web Services NOW!
at a One-Day Tutorial... Coming to a City Near You!



tem's default. As an example, if you're on a Windows NT server you'll see a time-out of 240 seconds, which is longer than reasonable to wait. The socket object has the ability to specify a time-out but this detail is hidden from you.

The solution to all three questions is to spin your own implementation of `URLConnection`. This really isn't that difficult. I recommend downloading the source from one of the open source implementations such as the Jakarta project and using this as a starting point. Once you have the source, you can create your own implementation that will allow you to pass in sockets and time-out values as well as your own `setRequestMethods`.



HOW DO I SORT MY VECTOR OF OBJECTS USING JAVA? DO I HAVE TO WRITE MY OWN SORT ALGORITHM?

A common question...and the short answer is no. Using the Collections API, introduced in Java 2, makes the whole process very easy. You achieve complete flexibility by implementing the Comparator interface.

```
Vector fileList = new Vector();
//-[ insert a bunch of cachedFile classes
```

```
//-[ Sorting by the hits
Collections.sort( fileList, new Comparator(){
    public int compare( Object o1, Object o2 ){
        if ( ((cachedFile)o1).hits > ((cachedFile)o2).hits )
            return -1;
        else
            return 1;
    }
});
```

```
//-[ Sorting on String
Collections.sort( fileList, new Comparator(){
    public int compare( Object o1, Object o2 ){
        return ( (String)(((cachedFile)o1)).names.compareTo
```

```
(String)(((cachedFile)o2)).names );
    }
});
```

```
class cachedFile extends Object {
    public String name;
    public int hits;
}
```

Conclusion

I would like to wish everyone a happy New Year and hope that you have enjoyed this column. I leave you with a quote by Herbert N. Casson:

"The people who succeed are the efficient few. They are the few who have the ambition and willpower to develop themselves."

...

Send your questions, praise, comments, and admiration to doctorjava@sys-con.com.

Published letters will be edited for length and clarity. Any reference to third parties or third-party products should not be construed as an endorsement by Doctor Java or Java Developer's Journal.

AUTHOR BIO

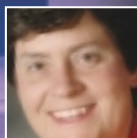
Doctor Java, a.k.a. James McGovern, moonlights as an enterprise architect with Hartford Technology Services Company, LLC. (www.htsco.com), an information technology consulting and services firm dedicated to helping businesses gain competitive advantage through the use of technology.

doctorjava@sys-con.com

Jump-start your Web Services knowledge Get ready for Web Services Edge East and West!

AIMED AT THE JAVA DEVELOPER COMMUNITY AND DESIGNED TO EQUIP ATTENDEES WITH ALL THE TOOLS AND INFORMATION TO BEGIN IMMEDIATELY CREATING, DEPLOYING, AND USING WEB SERVICES.

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASE TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI, AND XML, AND MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS, AND REMOTE REFERENCES.



PRESENTERS...

Anne Thomas Manes, Systinet CTO, is a widely recognized industry expert who has published extensively on Web Services and service-based computing. She is a participant on standards development efforts at JCP, W3C, and UDDI, and was recently listed among the Power 100 IT Leaders by Enterprise Systems, which praised her "uncanny ability to apply technology to create new solutions."



Zdenek Svoboda is a Lead Architect for Systinet's WASP Web Services platform and has worked for various companies designing and developing EXCLUSIVELY SPONSORED BY



systinet

BOSTON, MA (Boston Marriott Newton)**JANUARY 29**
WASHINGTON, DC (Tysons Corner Marriott)**FEBRUARY 26**
NEW YORK, NY (Doubletree Guest Suites)**MARCH 19**
SAN FRANCISCO, CA (Marriott San Francisco)**APRIL 23**

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE \$495 REGISTRATION FEE.

Register at www.sys-con.com or Call 201 802-3069

**i-TECHNOLOGY
EDUCATION**

THE CONTROVERSIAL "ROSS REPORT" ON MICROSOFT & .NET

Introduction by Alan Williamson

LATE LAST YEAR, in a move that many Java developers at the time likened to the wily fox inviting the unsuspecting hen round for a candlelit dinner, Microsoft Corp extended an invitation to JavaLobby founder Rick Ross to come to its headquarters...and check out .NET firsthand.

The invitation was the direct result of a public challenge the notoriously MS-skeptical Ross had issued to Microsoft in October. Ross promptly accepted the invitation. His resulting report sparked what the astonished founder describes as "more feedback, period, as well as more positive feedback, than any other item that's ever been published on JavaLobby's site."

So, has MS "brainwashed" Ross? The First JavaLobbyist himself, who is by no means an unsuspecting hen but on the contrary a fiercely independent thinker with a huge passion for Java development and developers, says not.

"Many of the people who have written postings to JavaLobby since my report was sent to members," explains Ross to *JDJ*, "were just reacting I think to the fact that I would even mention .NET...but they seem somehow to have missed my explicit message, right at the beginning of the whole Report, that .NET in my view represents what it always has and always will represent: a 'platform lock-in to Windows.'"

"Ignorance about .NET within the Java community," concludes Ross, "is much more dangerous than informed awareness."

True to our tradition of letting developers decide issues for themselves based on the facts, the editorial board of *JDJ* has resolved to let readers of *JDJ* judge for themselves.

Alan Williamson
Editor-in-Chief, *Java Developer's Journal*

HIGHLIGHTS OF RICK ROSS'S VERDICT ON .NET

Rick Ross Founder, JavaLobby • www.javalobby.org

THE .NET "MASTER BRAND" touches every one of Microsoft's business units, and the company appears to be more organized, aligned and excited than I have ever seen it.

The developer tools they have integrated into Visual Studio .NET are genuinely powerful and attractive, but the price is the same as it has always been – platform lock-in to Windows. Microsoft is actively working through ECMA to standardize the CLR, C#, and much of the .NET framework – a path leading ultimately to ISO. This ECMA effort may be primarily symbolic, however, since only a player with enormous resources and funding could possibly implement the standard. If you use .NET you can expect to be using it only on Windows for a long time to come. Even on Windows alone the .NET platform will be a formidable economic force that will eventually touch most of us, if not all of us.

The Strategic Value of Java Developers

Microsoft may be starting to remember that it needs to love developers, and Java developers are in the sweet spot – very frequently controlling decision-making power and influence over the distributed enterprise apps on which .NET's success will depend. There are signs that Microsoft may wish to renew good relations with the deeply alienated Java developer community, but cleaning up the mess it has created over the past few years would take amazing diplomatic skill and require concessions that I doubt Microsoft is able and willing to make. Certainly nothing could alter the cold rivalry between Microsoft and Sun, but things could get very interesting if Microsoft somehow found a way to win back a modicum of trust and respect from Java developers. Could Microsoft possibly be that smart? What would it take? It's a scenario that is hard to even imagine...

Observations & Opportunities

In many ways I think developers and consumers should welcome the emergence of .NET as a powerful competing force that will definitely keep the pressure on Java technology to adapt, evolve and improve. In fact, .NET will even provide new market opportunities for Java-based Web services since they are so easy to use within VB.NET and ASP.NET. Web services could be a critical factor in the mix, since the XML-based technology reduces the pressure for people to code in the same language in order to work together cooperatively. Business is business, so if Java developers can reasonably expect to profit from interoperating with .NET, then many will probably at least be willing to listen.

Faces of Real People, Not the Death Star

Microsoft is a most gracious host, and I was genuinely impressed by the intensity and intelligence of the people who presented the enormous set of products and technologies they will market under the .NET "master brand". These individuals were organized, informed, patient, thoughtful and non-defensive – people who were easy to like and whom I am glad to know. As one JL member wrote to me before my trip, the walls in Redmond are not painted black like the Death Star.

...With such excellent Java experts on hand the conversations were dynamic, even exciting. The questions we fielded and the answers we received were articulate. It's fair to say that everyone learned a lot...

The Agenda Was Jam-Packed

The days were packed with back-to-back sessions on the Common Language Runtime (CLR), Visual Studio .NET, C#, ASP.NET, ADO, the .NET framework classes and APIs, security, mobile development, and of course, Visual J# .NET...

I was captivated by the presentations by Jim Miller on the CLR and by Anders Hejlsberg on C#....

The Tools That Might Have Been

One of my major regrets about Microsoft's alienation of the Java world has always been that their fantastic talents and resources were not focused on advancing the state of the art in Java development tools...

Visual Studio .NET – The Tools That Are

Well, the answer is probably located somewhere within the Visual Studio .NET product. It integrates the design, development and testing experience for the more than 20 programming

wirelessEDGE
conference & expo



Plan to Exhibit

Provide the Resources To Implement Wireless Strategy

The conference will motivate and educate. The expo is where attendees will want to turn ideas into reality. Be present to offer your solutions.

INTERNATIONAL WIRELESS BUSINESS & TECHNOLOGY CONFERENCE & EXPO

CONFERENCE & EXPO
CONFERENCE & EXPO



Shaping Wireless Strategy for the Enterprise

Santa Clara, CA

May 7-9, 2002

Wireless Edge will provide the depth and breadth of education and product resources to allow companies to shape and implement their wireless strategy. Developers, i-technology professionals and IT/IS management will eagerly attend.

WHO SHOULD ATTEND

Mobile & Wireless Application Professionals who are driving their enterprises' wireless initiatives:

- Program Developers
- Development Managers
- Project Managers
- Project Leaders
- Network Managers
- Senior IT and Business Executives

Plan to Attend the 3-DAY Conference

Conference Tracks

| Track One: Development | Track Two: Connectivity | Track Three: Wireless Apps | Track Four: Hardware | Track Five: Business Futures |
|-----------------------------|-------------------------------|----------------------------|-------------------------------------|----------------------------------|
| WAP | Smart Cards | Education | Cell Phones/ | Wireless in Vertical Industries |
| i-Mode | Wireless LANs incl. Bluetooth | Health Care | World Phones | The WWW |
| Bluetooth / 802.11 | UMTS/3G Networks | Entertainment | PDAs | Unwired Management |
| Short Messaging | Satellite Broadband | Transport | Headphones/ Keyboards / Peripherals | From 3W to 4W: Issues and Trends |
| Interactive Gaming | | Financial Services | Transmitters/ Base Stations | "Always-On" Management |
| GPS / Location-Based | | Supply Chain Management | Tablets | Exploiting the Bandwidth Edge |
| Wireless Java | | | | Unplugged Valueware |
| XML & Wireless Technologies | | | | Wireless Sales & Marketing |

FOR EXHIBIT & SPONSORSHIP INFORMATION PLEASE CALL

201 802-3004

SPEAKER PROPOSALS INVITED

WWW.SYS-CON.COM

WWW.WIRELESSEGE2002.COM

SHAPE YOUR WIRELESS STRATEGY... SAVE THE DATES!



EXCLUSIVE SPONSORSHIPS AVAILABLE

Rise above the noise. Establish your company as a market leader. Deliver your message with the marketing support of



languages and the numerous frameworks that comprise the .NET platform. VS .NET is a “tour de force” powerhouse, and it is readily extensible by third-party tool providers. There are already commercial add-ins coming online, and the VS .NET product isn't even officially launched yet. It appears the development experience is more integrated in VS .NET than ever before. It was impressive how many different tools and capabilities were demonstrated by the various presenters, all within the scope of the comprehensive master program....

An Integration Opportunity?

I wonder if someone will see a market opportunity for integrating Jikes, ANT, and the real JDK into this environment? It certainly seemed like some Java integration would be feasible and even easy, but it would obviously take more exploration to understand what the potential benefits would be? ... Under the right circumstances integrating VS .NET and Java could be a very interesting project.

Visual J# .NET – Not What We Thought

Visual J#.NET is an intriguing product, if somewhat enigmatic within a larger view of .NET as a platform. It turns out that Microsoft completely botched the initial “leak” about J# and JUMP to the media, positioning it as a tool to “help” Java programmers convert their Java code to C#. That was a foolish marketing mistake that will probably forever discolor the reality that J# is a much more sincere effort at providing first-class Java support (up to a point) within .NET. J# is not a code converter at all, but rather an implementation of nearly the entire JDK1.1.4 platform inside of .NET. I suppose JDK1.1.4 is the point at which Microsoft's rights terminate under the settlement of the Sun lawsuit...

Compatible, In That Microsoft Kind of Way

The interesting thing about Visual J# .NET is that Microsoft claims it passes the full 15,000 tests of the Java conformance testing suite. I don't know how it could do this without RMI and JNI in there? Even if you exclude the tests involving those features, however, it would still appear that Microsoft has invested

significantly in creating a Java implementation for .NET that is as conformant as they could make it. We watched an applet with AWT graphics compiled, deployed, and executed by J# within the .NET runtime, which was pretty cool. ...

Why Make Visual J# .NET Anyway?

So why did Microsoft create Visual J# .NET? I mean, it took a lot of money and effort to provide this much support for the Java programming language and platform within the .NET frameworks... The stock answer is to provide support for VJ++ users, but the answer posed by Gary Cornell may well provide the best explanation: Java is the language now used by nearly all academic programming courses. If Microsoft does not have support for Java, as a language, within its new developer tools, then it is sure to have a very difficult time gaining acceptance for Visual Studio.NET within academia... because Java is the language they now use to teach “Programming 101.”...

My Conclusions, for the Moment

In conclusion, the .NET platform is huge, and we will all probably encounter it in one form or another. It seems to me that ignorance about .NET within the Java community is much more dangerous than informed awareness. I realize that it's only natural for Java developers to consider Microsoft's offerings to be suspect, but I think we should not close the door on .NET blindly. There will soon be a lot of discussion about the comparative pros and cons of J2EE and .NET, and J2EE won't win by default just because .NET originates from Microsoft. Interestingly, there may even be some excellent opportunities for the Java world and the Microsoft world to interoperate profitably via XML Web services. If the USA and China can have healthy economic trade despite significant ideological differences, then there's a possibility that those of us in the freedom-loving Java world can engage in healthy economic trade with the many millions who will be locked into Windows and .NET. Stranger things have happened... ●

To read Ross's “My December Trip to Microsoft” in full, go to www.sys-con.com/java/.

READER RESPONSES: THE ROSS .NET REPORT DIVIDES JAVA DEVELOPERS

Rick Ross sold out!

Posted by <mailto:mihir@usa.net>Mihir Karia (mihir@usa.net) 2001-12-20 16:59:22

I suspect that Rick works for MS and that they (MS) are once again attempting to sabotage the Java platform. Since they cannot do it openly, they are doing it covertly.

There are so many issues that Rick could have brought up about the .NET platform but instead his article reads like a “Ooh! .NET is not so bad after all.” Give me a flying break!

He wonders why MS introduced J# into the .NET environment. Yeah right! It becomes pretty obvious as he later writes “They have also added extensions to J# so that people can use Java more effectively in conjunction with inline metadata and other features of .NET.” Has he missed the point of Java and platform independence? A Java lobbyist my rear!

If we have people like Rick Ross lobbying for Java, we are surely doomed.

December in Redmond

Posted by <mailto:jt@iwaypublishing.com> John Thompson (jt@iwaypublishing.com) 2001-12-20 20:28:20

Nice work on the .NET report. In this age of interoperation, too many Java developers take far too much pride in their ignorance of all things Redmond.

You are correct, Java developers lost a lot when J++ development was curtailed in '97 – no one builds GUIs like Microsoft. No one tests usability, no one leverages Windows, no one completes the key-chord implementations,...like they do.

Also, as you mention, it is clear that there are some things lacking in the Java core language. Microsoft was putting some of those things in – against the spirit of the write-once run-anywhere mantra, but still valuable features. The fact that the response to that behavior was to curtail any enhancements at all is a loss to us

all. After all, C++ continued to evolve for many years with new and powerful features.

I can think of several omissions in Java that hamper enterprise development. For example (and I know this is controversial), the lack of a preprocessor makes developing and coding applications larger than the original-vision wizzy applet very trying. Microsoft had added this.

The threading model in Java is convenient in its ubiquity, but very limited in scope. There's room for growth here.

Not to knock it all – we're all getting a lot done with what's there!

And, right, if China and the US, if Russia and the US can work together, we'd all better know something about .NET.

Rick, you've made a welcome contribution to that learning process.

To read the full text of these and other responses, or to add your own comments, go to www.sys-con.com/java/.

**Special
online offer**

**PICK
4**

Subscribe

for
One
Special Low
Price

Wireless Business & Technology
Java Developer's Journal
Web Services Journal
XML-Journal
WebLogic Developer's Journal
WebSphere Developer's Journal
ColdFusion Developer's Journal
PowerBuilder Developer's Journal

**SYS-CON
MEDIA**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

WWW.SYS-CON.COM

EnginData Presents

2002 Developer Market Survey Reports

\$2,995

Java
Developer
Market Survey
2002

\$2,495

Web Services/XML
Developer
Market Survey
2002

\$1,995

Wireless
Developer
Market Survey
2002

Our comprehensive reports offer insight and strategy to guide your most critical business decisions in today's fastest growing technologies...

- ✓ Establish your product and marketing strategy
- ✓ Understand your customers' needs
- ✓ Evaluate technology and trends

Preview and order reports at www.engindata.com

engindata.com

engindata ✓
RESEARCH

DOT.COM

- buyer's guide
- java forums
- mailing list
- java jobs
- java store

MAGAZINE

- advertise
- authors
- customer service
- editorial board
- subscribe

CONTENT

- archives
- digital edition
- editorial
- features
- interviews
- product reviews
- source code



plugged in (in)

THE JAVA DEVELOPER CONNECTION™ PROGRAM

Get pre-release Java™ technology downloads

REGISTER TO GET THEM

 **Sun**

JAVA take it to the next

What's Online... January 2002

JDJ Online

Visit www.javadevelopersjournal.com every day for fast-breaking Java news and events. Know what's happening in the industry, minute by minute, and stay ahead of the competition.

2002 Readers' Choice Awards

Vote for your favorite Java software, books, and services in our annual **JDJ** Readers' Choice Awards. Categories include Best Java Testing Tool, Best Mobile Database, Best Java Messaging Tool, Best Java E-Business Framework, Best Java EAI Platform, Best Java Web Services Development Toolkit, and Best Database Tool/Driver.

The **JDJ** Readers' Choice Awards program, often referred to as the "Oscars of the software industry," has become the most respected industry competition of its kind. Winners will be announced in June at Web Services Edge 2002 Conference and Expo in New York City.

JavaDevelopersJournal.com Developer Forums

Join our new Java mailing list community. You and other IT professionals, industry gurus, and **Java Developer's Journal** writers can engage in Java discussions, ask technical questions, talk to vendors, find Java jobs, and more. Voice your opinions and assessments on topical issues – or hear what others have to say. Monitor the pulse of the Java industry!

Search Java Jobs

Java Developer's Journal is proud to offer an employment portal for IT professionals. Get direct access to the best companies in the nation. Learn about the "hidden job market" and how you can find it. If you're an IT professional curious about the job market, this is the site to visit.

Simply type in the keyword, job title, and location and get instant results. You can search by salary, company, or industry.

Need more help? Our experts can assist you with retirement planning, putting together a resumé, immigration issues, and more.

JavaBuyersGuide.com

JavaBuyersGuide.com is your best source anywhere, anytime on the Web for Java-related software and products in more than 20 mission-critical categories, including application servers, books, code, IDEs, modeling tools, and profilers. Check the Buyer's Guide for the latest and best Java products available today. ☉




WAKE SOFT
Learn more now!



Click for a FREE 30-day trial
AltoWeb



M-1 Mobile App Server
aligo
Download Now!



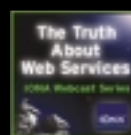
sitiraka



Get SonicMQ™
sonic SOFTWARE



HERANT
DataDirect



The Truth About Web Services
What Webcast Is In



KADA



spiritsoft
going beyond jms



Install Anywhere
PowerUpdate
ZEROG
www.ZEROG.COM
Click here to try

SYS-CON Media, the world's leading publisher of i-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebSphere.



WebSphereDevelopersJournal.com

WebSphere
DEVELOPER'S JOURNAL



Introductory Charter Subscription

SUBSCRIBE NOW AND SAVE \$31.00 OFF THE ANNUAL NEWSSTAND RATE

ONLY \$149 FOR 1 YEAR (12 ISSUES) REGULAR RATE \$180

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Do You Have Access to the Internet?

The World's Leading Independent WebSphere Developer Resource

Then Subscribe Online and Save \$31!

It's that easy

SHOP ONLINE AT **JDJSTORE.COM** FOR BEST PRICES OR CALL YOUR ORDER IN AT **1-888-303-JAVA**

BUY THOUSANDS OF PRODUCTS AT GUARANTEED LOWEST PRICES!

GUARANTEED BEST PRICES FOR ALL YOUR **WEB SERVICES** SOFTWARE NEEDS



CAPE CLEAR

\$445.00

CapeStudio

CapeStudio is a Rapid Application Development (RAD) environment for creating Web Services and XML-based applications. It consists of an integrated toolset. CapeStudio Mapper enables XML data transformations in a graphical environment and generates standard XSLT. CapeStudio WSDL Assistant generates Web Services client proxy and server skeleton code based on standard WSDL. UDDI Registry Browser searches for available Web services descriptions.



ALTOWEB

\$3,540.00

Application Platform Release 2.5

The AltoWeb Application Platform lets you build, deploy, and manage J2EE applications and Web Services up to 10x faster without requiring extensive J2EE or Web Services expertise. How? By replacing lengthy, custom and complex J2EE, XML and Web Services coding with rapid component assembly and reuse.



CAPE CLEAR

\$950.00

CapeConnect Three

CapeConnect is a complete Web Services platform that allows you to automatically expose your existing Java, EJB and CORBA components as Web Services without writing code. CapeConnect is capable of connecting a wide range of technologies (including Java, J2EE, CORBA, C++, COM, C#, Visual Basic, and Perl) using Web Services standards such as SOAP, WSDL, and UDDI. CapeConnect can be used internally over intranets and also can be used to expose business logic over the Internet, for use by customers and partners.



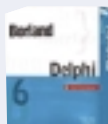
BORLAND

221980 \$2,999.00

Sale Price \$2,849.99

Delphi

Delphi 6 makes next-generation e-business development with Web Services a snap. BizSnap Web Services development platform simplifies business-to-business integration by easily creating Web Services. DataSnap Web Services-enabled middleware data access solutions integrate with any business application. Check out our Web site for system requirements.



SITRAKA

\$3,999.00

Sale Price \$3,799.00 JClass ServerChart V1.1 Bytecode

If you've used JClass Chart in your client-side applications, you already know the value of data visualization to your end-users. With JClass ServerChart, you can bring a wide range of charts and graphs right to their Web browsers, giving them real-time access to business-critical data in an intuitive environment.



SILVERSTREAM

\$495.00

Extend Application Server Developer Edition (5 User)

SilverStream eXtend allows you to rapidly create, assemble, consume and deploy Web Services. It includes an advanced and intuitive development environment. This workbench provides everything for developers and business analysts to create, assemble, consume and deploy services. And it integrates with your favorite code editors and source control systems.



Infragistics Launches JSuite 6.0 (Cranbury, NJ) – Infragistics announced the release of JSuite 6.0, an all-inclusive suite of presentation-layer Java components that offer developers a complete Java solution.



Infragistics simultaneously announced it would launch PowerChart Server Edition, a highly scalable Java server-side charting component, as a standalone product. www.infragistics.com

Sun Releases Java BluePrints for Wireless Program

(Santa Clara, CA) – Sun Microsystems, Inc., announced the immediate availability of their Java BluePrints for Wireless program, a collection of end-to-end best practices, guidelines, and architectural recommendations that demonstrates how to build a distributed, transaction-oriented enterprise application using J2EE on the server and J2ME on the client.



The new Java BluePrints program is delivered, free of charge, through white papers and the Java Smart Ticket, a real-world

sample application that illustrates various recommended development techniques. It's available from Sun's Web site at <http://java.sun.com/blueprints> and from the Wireless Developer Initiative site at <http://developer.java.sun.com/developer/products/wireless>.

Rational Announces Rational Suite v2002

(Lexington, MA) – Rational Software announced the release of Rational Suite v2002, highlighted by a new product, Rational ProjectConsole.



Rational Suite v2002 comes with 15 pretested, integrated, and co-released products in eight Rational Suite editions. New features include Java enhancements, two new runtime observation capabilities, and Rational ProjectConsole, a Web-based tool that helps

software development teams measure project progress and software quality. www.rational.com

Codagen Releases Gen-it v2.0

(Montreal) – Codagen Technologies Corp. announced the general availability of Gen-it v2.0,

an add-on enhancement software solution that integrates with modeling and IDE tools to generate up to 100% of the code associated with any application architecture, and up to 70% of the total application code. www.codagen.com



Together ControlCenter Earns IBM ServerProven Validation

(Raleigh, NC) – TogetherSoft Corporation's Model-Build-Deploy Platform for end-to-end software



development – Together ControlCenter – has been validated as an IBM ServerProven solution.

TogetherSoft received ServerProven approval after completing the validation program at the IBM Solution Provider Center in Waltham, Massachusetts. There, Together ControlCenter was installed and tested on four IBM eServer xSeries 330 running Microsoft Windows NT, Windows 2000, Red Hat Linux, and SuSE Linux. www.pc.ibm.com/ww/eserver/xseries/serverproven/index.html www.togethersoft.com

TechMatrix to Distribute Aligo M-1 Mobile App Server

(Tokyo and San Francisco) – TechMatrix has entered an agreement with Aligo, Inc., to be the first Japanese distributor of the Aligo M-1 Mobile Application Server.



Under the terms of this agreement, TechMatrix can now offer its customers Aligo's Java-based mobile application server software, which is suited for companies that want to make corporate data and applications available to multiple wireless devices. www.aligo.com www.techmatrix.co.jp



SYS-CON Events Announces Web Services Edge 2002 World Tour Tutorial Series (Montvale, NJ) – SYS-CON Events, Inc., announced the dates and locations of a new tutorial series in preparation for SYS-CON's Web Services Edge 2002 International Web Services Conference & Expo. The first leg of its world tour, "Developing SOAP Web Services," is exclusively sponsored by Systinet Corporation.

An intense one-day tutorial aimed at the Java developer community, "Developing SOAP Web Services" is designed to equip professional developers with all the tools and information they need to immediately begin creating, deploying, and using their own Web services. Expert practitioners will take an applied approach and cover base technologies such as SOAP, WSDL, UDDI, and XML, as well as more advanced topics such as security, J2EE integration, exposing legacy systems, and integrating Web services into an existing IT infrastructure.

Web Services Edge 2002 East will be held June 24–27 at the Jacob Javits Convention Center in New York, and Web Services Edge 2002 West will be held October 1–3 at the San Jose Convention Center in California.

The one-day Web services tutorial kicks off in Boston on January 29, then travels to Washington, DC, New York, and San Francisco. Further information and online registration is available at www.sys-con.com.



Java Developer's Journal Ranked First in Survey

(Montvale, NJ) – Evans Data Corporation has ranked *Java Developer's Journal* as "the most trusted developer publication" among developers who use Java, according to SYS-CON Media, the world's leading *i*-technology publisher. The research results were published in the Evans Data Developer Marketing Patterns 2001 Annual Report, an independent market research report prepared by the leading market research firm serving software and developer markets.

"We are very pleased to see, in our sixth year of publication, that *Java Developer's Journal* continues to serve the fast-growing Java developer community as the hands-down leader of quality Java information in the world. *JDJs* unmatched leadership is not based solely on its circulation – which is larger than all other Java publications put together – but also on the high quality of its editorial content," said Alan Williamson, editor-in-chief of the magazine. "The Evans Data report confirms what we hear from our readers and *JDJs* sponsors and advertising partners every day."

www.sys-con.com



Once you're in it...

- * Wireless Business & Technology
- * Java Developer's Journal
- * XML Journal
- * ColdFusion Developer's Journal
- * PowerBuilder Developer's Journal

reprint it...



Contact Carrie Gebert
201 802-3026
carrieg@syz-con.com



Re Prints

Your Own Magazine

- Do you need to differentiate yourself from your competitors?
- Do you need to get closer to your customers and top prospects?
- Could your customer database stand a bit of improvement?
- Could your company brand and product brands benefit from a higher profile?
- Would you like to work more closely with your third-party marketing partners?
- Or, would you simply like to be a magazine publisher?

SYS-CON Custom Media is a new division of SYS-CON, the world's leading publisher of Internet technology Web sites, print magazines, and journals.

SYS-CON was named America's fastest-growing, privately held publishing company by *Inc. 500* in 1999.

SYS-CON Custom Media can produce inserts, supplements, or full-scale turnkey print magazines for your company. Nothing beats your own print magazine for sheer impact on your customers' desks... and a print publication can also drive new prospects and business to your Web site.

Talk to us!

We work closely with your marketing department to produce targeted, top-notch editorial and design. We can handle your distribution and database requirements, take care of all production demands, and work with your marketing partners to develop advertising revenue that can subsidize your magazine.



So contact us today!

East of the Rockies,
Robyn Forma,
robyn@syz-con.com,
Tel: 201-802-3022

West of the Rockies,
Roger Strukhoff,
roger@syz-con.com,
Tel: 925-244-9109

Premiering...this winter

subscribe Now!

FORFAST DELIVERY

Go Online and Subscribe Today!

The World's Leading
Independent WebLogic
Developer Resource

FOR WLS DEVELOPERS BY WLS DEVELOPERS

WebLogic BEA DEVELOPER'S JOURNAL

Helping you enable inter-company collaboration on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks and more!

SPECIAL INTRODUCTORY OFFER
SAVE \$31*
HURRY, DON'T DELAY! OFFER EXPIRES DECEMBER 31, 2001

WebLogicDevelopersJournal.com

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebLogic.

*Only \$149 for 1 year (12 issues) regular price \$180.



When the Going Gets Tough . . .

. . . the tough work smart



WRITTEN BY
BILL BALOGLU &
BILLY PALMIERI

According to our sources and associates, this is the toughest job market that anyone in the IT industry has seen in a long, long time.

Unless you've been living in a cave for the past five years, you already know the story. After unprecedented growth and feverish hiring across the spectrum of high-tech industries, the party is over.

Start-ups have gone belly-up, and as the sluggish economy turns various shades of grim, even the industry's biggest, most "secure" companies are in rapid downsizing mode.

We've addressed various aspects of this change in several columns over the past year. But now that even the most senior of engineers are struggling to find work, effective job search techniques are more critical than ever.

Fact: Since unemployed tech professionals far outnumber available jobs, employers who post an open position no longer have to scramble to fill those positions. They must now sort through the hundreds of résumés that flood their in-boxes.

How do you stand out from the crowd or even get your résumé seen in this sudden flood of competition?

How do you get the attention of the hiring people to get an interview?

Here are a few dos and don'ts for finding work in this tight market:

Do:

1. **Network** – contact everyone you've ever had a good working relationship with (former managers, peers, and team members) and let them know you're available. Send them an updated résumé and ask them to pass it on to potential hiring managers at their company.

Most employers know that the best candidates are still referrals from valued employees.

Even if there's no position immediately available, try to set up a brief informational interview to introduce yourself and find out what they're doing. This gives potential employers a chance to match your face to your résumé – and keep you in mind when positions open up.

2. **Attend as many industry networking functions as you can.** Bring a fresh

batch of résumés with you and collect as many business cards as you can. Keep in e-mail touch with new contacts on a regular (monthly) basis to refresh their memories and let them know you're still available.

3. **Re-edit, highlight, and target your résumé for every job you apply for.** This is time-consuming but effective.
4. **Research the company you're applying to and contact anyone you know who may have worked there at any time.** They could help direct you to the right group or manager who may need someone with your abilities.
5. **Understand current market conditions and be prepared to make adjustments to your rate and/or expected level of seniority.** You may have been a senior engineer last year but now you may be considered intermediate. If you want to work, keep your options open to part time or contracting positions – whatever it takes. Be flexible.
6. **Work with reputable agencies to be considered for positions that aren't posted publicly.** Many companies avoid sorting through the deluge of résumés by working directly with agencies.
7. **Keep on top of new posted positions on a daily basis.** Your odds of standing out in a crowd of 300 résumés are not great, but they're better if yours is one of the first 20 résumés submitted. *Remember:* Your time and your timing are critical. Work smart and act quickly.

Don't:

1. **Set up a system that automatically sends out a generic résumé to every job posting that contains buzz words you're interested in.**

Sending out the same résumé for different types of jobs suggests that either you don't understand what they're looking

for (or what the company is doing) or you don't really care. Also, good managers and recruiters remember résumés and names. They may end up skipping over you for the right position because you sent in your résumé earlier for the wrong one.

2. **Assume the only open jobs are the ones posted on the Web or in the newspaper.** While you need to stay on top of these openings, and apply to as many of them as you're qualified for, networking contacts is always best.

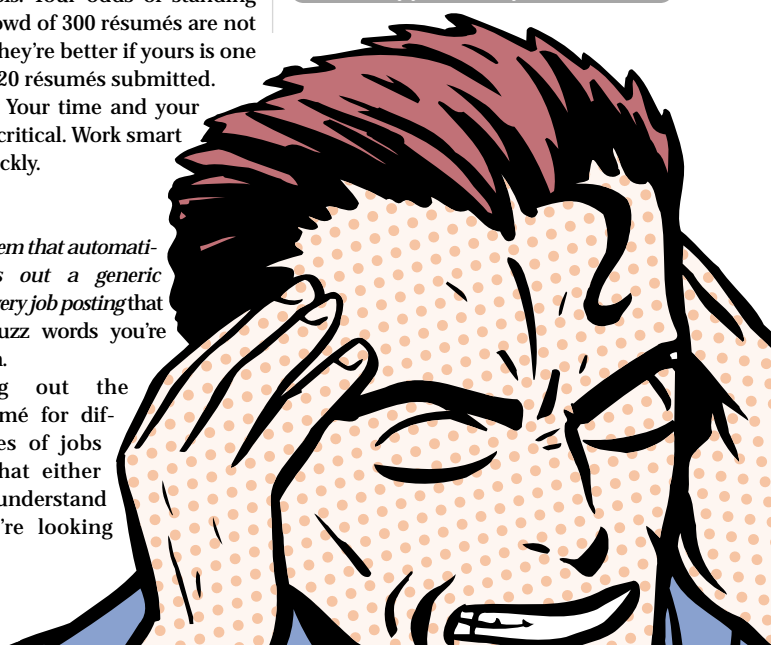
3. **Get discouraged.** While industry leaders agree that we may never again see the kind of rapid growth and massive hiring of the past few years, they also agree that things will turn around.

The high-tech industry may ultimately evolve into a slightly different animal, but for those who stay flexible, determined, and thorough in their job search, there will always be opportunities.



What have your job search experiences been like in the past few months? Let us know what challenges you've faced and what techniques have worked for you. E-mail us at jdjcolumn@objectfocus.com.

jdjcolumn@ObjectFocus.com



sys-con media

www.sys-con.com

Next Month



GETTING STARTED WITH JAVA ON THE IPAQ

Build and deploy Java apps
by Rob Tiffany

STEPPING OUT OF THE JAVA DATA OBJECTS CLOSET

Advanced issues using JDO
by Yaron Telem and Shay Litvak

BOOK REVIEW

Java Internationalization
by Andrew Deitsch, David Czarnecki, and Andy Deitsch
reviewed by Ajit Sagar

CAREER OPPORTUNITIES

"How's the Boss?"
by Bill Baloglu and Bill Palmieri

PATTERN FOUNDATIONS: THE OPEN-CLOSED PRINCIPLE

Basic patterns
by Kirk Knoernschild

JAVA DEVELOPERS
JOURNAL



www.wbt2.com

SUBSCRIBE NOW

www.javadevelopersjournal.com

TO THE

www.xml-journal.com

FINEST

www.coldfusionjournal.com

TECHNICAL

www.powerbuilderjournal.com

JOURNALS

www.webspheredevelopersjournal.com

IN THE

www.wldj.com

INDUSTRY!

www.wsj2.com



subscribe online www.sys-con.com or call 800 513-7111

**SYS-CON
MEDIA**

wireless | java | xml | coldfusion | powerbuilder | websphere | weblogic | web services

The Search for Beauty



WRITTEN BY
BLAIR WYMAN

“How did I get here?” There’s a question I’ve asked myself many, many times over the years. Decorum prevents me from recounting all the contexts to which “here” has referred, but suffice it to say that my inflection has become less frenzied as I matured.

I suppose the logical context to which “here” refers this time would be my chosen career as a computer programmer. Of all the possible directions I could have taken – or been forced to take – somehow I ended up working at something I really enjoy, making a decent living in the process. By any measure I count myself among the supremely fortunate.

Of course, it hasn’t always been so. Despite every generation’s dubious claim to its original discovery, “teen angst” was certainly alive and well in the ’70s when I was a teen. Had I known what to call it back then, I might have applied for “poster child” status.

I guess I was a pretty “smart” kid (though most of my teachers would probably choose to narrow that characterization to selected parts of my anatomy), and I got fair grades without trying very hard. I had a good memory and could usually absorb enough information to pass those simple, public-school tests by a goofy sort of psychic osmosis. Was I lucky? I’m not sure. Perhaps I’d have learned more if I had to work harder. It’s definitely a conundrum.

While I barely graduated high school (who knew foosball wasn’t a for-credit course?), I did so with a sufficiently high grade-point average to graduate with “honors.” To this day I remember being earnestly chastised at the graduation ceremony by a hard-working classmate who had just missed the cut. “You didn’t earn that,” she fumed, pointing at my gold tassel. I just sort of smirked, if I remember correctly, though her words cut me as deeply as only the truth can.

Are brains (which I pretend to have) or beauty (which, if I ever had, I have hence given to my children) valid causes for pride? Personally, I don’t think so. After all, capable brains and natural beauty are both simply accidents of birth, not the result of any overt acts. In my idealistic view, pride is something that should be earned. At least that’s the belief I try to instill in my smart, beautiful children – the ones I’m so deliriously proud of.

What lured me into the realm of making computer science my livelihood? To some extent computer science was an accidental direction in my life. After my cab-driving years, when I went back to college to finish some sort of degree, I found that math was far and away my favorite gig. The wee bits of memory capability that had survived my youth seemed curiously well suited to the mysteries and arcana of mathematics: I’d found my “thang.”

As I progressed through the undergraduate curriculum, a weird sort of appreciation bloomed in me: an appreciation for the mysterious beauty and surprising consistencies of so-called “higher” mathematics. Even at this relatively introductory level, the logically unifying connections between previously unrelated topics in math seemed to manifest themselves as stunning new aspects of the “natural” world. Certainly sunsets and flowers can be breathtaking, but who would have thought that the same could be said of systems of equations or elegantly crafted proofs?

Just before I got my math degree, my father died rather suddenly. He was about

to start his thirtieth year teaching engineering graphics at the local college; then he was gone, leaving my Mom on her own. After the joy I had experienced discovering mathematics, I knew I wanted to pursue a postgraduate education, and now I simply had to do so at my Dad’s old school so I could be close to home.

However, at that time Dad’s college didn’t have a postgraduate offering in math. I remember sitting at the dining room table, fanning the school’s catalog and literally closing my eyes and stabbing my finger at the first open page. “Hmmm...computer science...” Something in my head clicked, and the rest is history.

Of course, there were times when I doubted myself, suffering a sort of “twenty-something angst.” My first semester, since my bachelor’s degree was not in computer science, I signed up for the entire undergraduate core of the curriculum – assembly language, data structures, and compilers – in one fell swoop. Passing these classes, while maintaining a part-time job to support my new family, almost did me in. But fortunately there was – and still is – a certain beauty to discover in computer science too. That search for beauty kept me going, and the discoveries were just beginning.

I’m still searching for and finding beauty. Part of the satisfaction I derive from writing these little bits of fanciful fluff is the belief that you, dear reader, appreciate beauty too. Your presumed attraction to Java – a beautiful language, indeed – is evidence enough for me. ♣

blair@blairwyman.com

