SYS-CON MEDIA

WHAT'S THE POINT OF **TEST-DRIVEN DEVELOPMENT?** pg. 8

# JAVA™ DEVELOPER'S JOURNAL

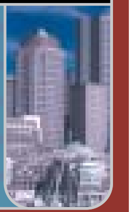
*The World's Leading Java Resource*

November 2003 Volume:8 Issue:11

www.JavaDevelopersJournal.com

**International Conference & Expo**

Edge 2004 EAST



Feb. 24–26, 2004
Boston, MA

*details on pg. 53*

RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2004

$5.99US $6.99CAN

## MIDI WITH JAVA

by Mike Gorman
page 42

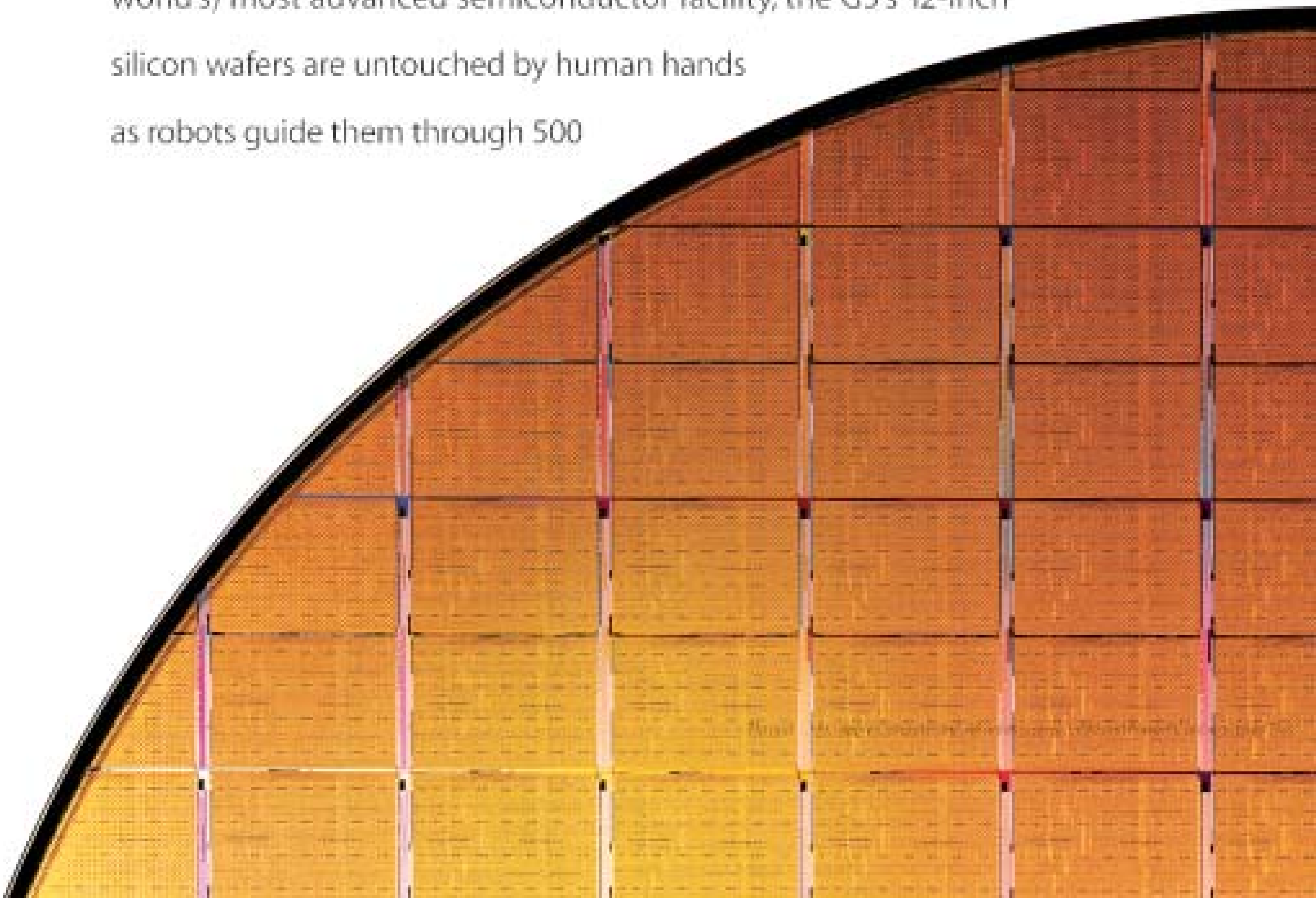
Get Optimized.™

# The dawn of a new PC era.

For the last decade, every processor chip in every personal computer in the world has been based on 32-bit architecture. It was the best technology we had. Until today. Introducing the revolutionary PowerPC G5 processor, the world's first 64-bit processor for personal computers.

*The world's most advanced personal computer chips are manufactured in the world's most advanced semiconductor factory.*

Before now, a chip this formidable could only be found in the world's fastest servers and supercomputers. Which is precisely where the G5 chip came from. Developed by IBM and Apple; the G5's DNA is from the core of IBM's highest-performance, 64-bit POWER4 processor. But just as impressive as the G5's pedigree is how it's manufactured. In IBM's (and the world's) most advanced semiconductor facility, the G5's 12-inch silicon wafers are untouched by human hands as robots guide them through 500

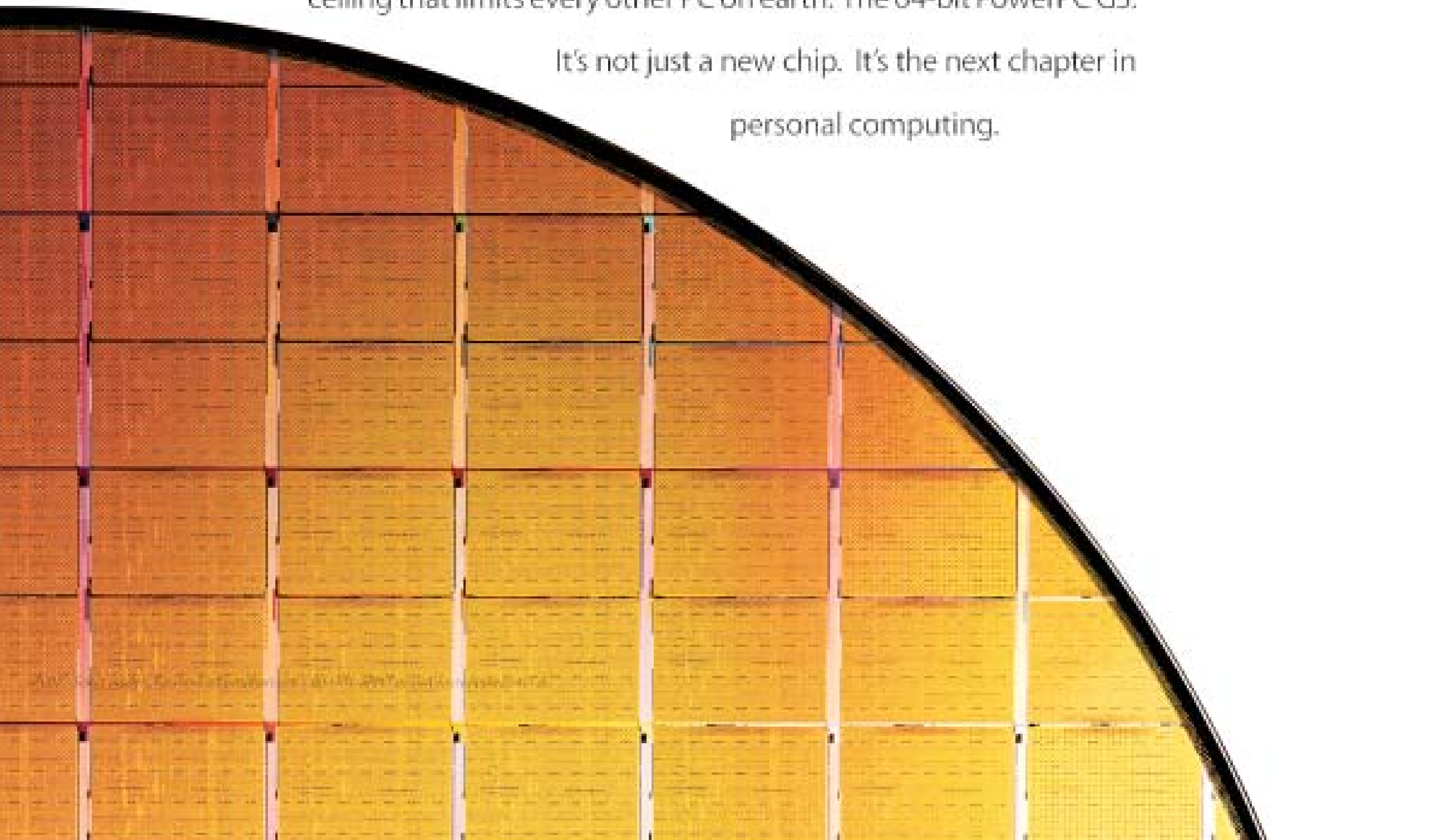


# The 64-bit processor.

processing steps, creating 58 million transistors and connecting them with over 1000' of copper wire that's less than 1/800th the width of a human hair.

The new PowerPC G5 has a 1-gigahertz frontside bus* that moves data in and out of the processor almost twice as fast as the competition, removing a key bottleneck that limits performance. And it can support more than 200 in-flight instructions at a time – 71% more than the 32-bit Pentium 4. Perhaps most importantly, the G5's 64-bit architecture can address dramatically more memory – over 4 billion times more than 32-bit chips – so that systems built around the G5 can shatter the 4-gigabyte memory ceiling that limits every other PC on earth. The 64-bit PowerPC G5. It's not just a new chip. It's the next chapter in personal computing.

*The world's first 64-bit desktop processor can only be found here: Inside the new Power Mac® G5, the world's most powerful personal computer.*



*[illegible]

Monitored

Maximized

JAVA DEVELOPER'S JOURNAL

**Editorial**

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

**Production**

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

**Writers in This Issue**

Jason Bell, Steve Close, Glen Cordrey,
Ashish Garg, Mike Gorman, Onno Kluyt,
Lucas McGregor, Dan North, Joseph Ottinger,
Kirk Pepperdine, Hendrik Schreiber, Alan Williamson

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

**Subscriptions**

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

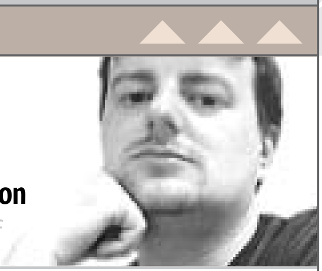
**Editorial Offices**

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

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

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

# Finally a Device That Delivers



**Alan Williamson**
Editor-in-Chief

I want a wireless handheld device that works with me and doesn't make me jump through hoops just because I want to use Java. I don't even want to know Java is running; I just want it to do its job and make my life easier.

I've been through a variety of phones and PDAs, and each one had major irritations. If it wasn't the poor performance it would be the clunky interface. Then there would be the balance of what the device wanted to be when it grew up: was it a PDA with a phone, or was it a phone with fancy features?

The closest device to date was the Sony P800. I blogged my whole experience with the beast and felt it fell far short of the marketing hype that accompanied it. I was beginning to lose hope that Java could deliver its promise on mobile devices.

Some good news to report: I believe I've found the ultimate Java device, and it's so good I'm devoting this entire editorial to it. The device is the BlackBerry 7xxx series.

I love it. It has everything you need from a handheld. Large color screen, long battery life, very light, always-on connectivity, and the killer feature: a small QWERTY keyboard that is easily used with your thumbs. The software bundled with it is strong and useful. The device has been well thought out from both a hardware and software point of view.

Screen quality can't be underestimated in these devices. It's the first thing that will sap the battery life away, and due care must be given to getting it just right. The screen is large and very clear and comes with a backlight facility that pops on should the light fade on you. The clarity makes using this device a joy.

There are so many features on the BlackBerry 7xxx that I could go on for pages. For example, a really nice touch is that it charges through the same USB cable you sync with. One less power cube to carry around.

If it's so great, why isn't everyone using it? Good question, and one I have been giving a lot of thought to. I think the blame lies primarily with RIM, the licensors of BlackBerry's wireless e-mail technology and, secondarily, with the wireless operators.

To know about the device you would first have to know someone or see someone using it. Assuming you know of it and head over to their Web site (www.blackberry.com), you could still be forgiven for being in the dark. This is one of the greatest Java devices ever, but you won't know that after you visit their site. RIM is not very good at marketing this device.

RIM is even poor at dealing with the press. I have repeatedly asked for information and interviews, and I'm still waiting. Why are they hiding?

This device, with its large screen and integrated keyboard, is crying out for Java developers to be let loose with it. This device will make you want to return to your IDE after dinner and start letting your creative juices flow. Devices like these could really secure Java's role in the mobile arena.

RIM is not the only one to blame. The wireless telcos aren't making it easy. I had to jump through hoops to get mine connected here in the UK, and even after that I'm being charged a fortune for data going in and out. Compare this to the "all-you-can-eat" data plans in the U.S. It shouldn't be this hard (or costly) surely?

Competition is hot on BlackBerry's heels. Palm, for example, has a very nice device with their Tungsten W model. Aimed at the same "always-on" market as the BlackBerry, it will be interesting to see how well they gain market- and mindshare. Their press folks have already contacted me, and they're doing a lot to convince Java developers to come to their device.

The future for these Java devices is looking rosy – if the manufacturer can get its act together and create the groundswell of development to allow the community to build the rich array of software it requires.

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

*alan@sys-con.com*

# Test-Driven Development Is Not About Testing

Dan North

I am always on the look out for good questions to ask candidates in an interview. Not the "How many oranges can I fit in this room?" kind of nonsense (the stock response to which is apparently "with or without us standing in it?"). Nor the picky, encyclopedic type such as "In the javax.obscure.DustyCorner class, which method throws a FullyDocumentedException?" (If you do not respond with "I would check the Javadocs" on the grounds that you actually know, you really ought to get out more.)

Instead, I like the sort of technical question that allows candidates to demonstrate real insight; where they can show not only technical depth and breadth, but also a mature understanding of the software development process. So I was delighted when a colleague offered me a perfect interview question, namely: "What is the point of test-driven development?"

Test-driven development (TDD) has grown out of the Agile software movement (www.agilealliance.org) and Extreme Programming (XP) in particular. Extreme Programming stipulates a set of best practices that collectively encourage core values such as feedback and simplicity. The feedback occurs in the form of tests, by delivering in short iterations, and by the simple expedient of talking to one another. The simplicity comes from the process of refactoring – ruthlessly – and from only delivering exactly what the software has to do right now.

Kent Beck, the original champion of XP, has extracted the essence of its development practices and named it test-driven development. And so to the model interview answer. The point of TDD is to drive out the functionality the software actually needs, rather than what the programmer thinks it probably ought to have. The way it does this seems at first counterintuitive, if not downright silly, but it not only makes sense, it also quickly becomes a natural and elegant way to develop software.

We start by writing some client code *as though the code we want to develop already existed* and had been written purely to make our life as easy as it could possibly be. This is a tremendously liberating thing to do: by writing a model client for our code, in the form of a test, we can define programmatically the most suitable API for our needs. In addition, we assert the behavior we want.

Obviously this won't even compile, and this is the counterintuitive part – the code that will sit on the other side of the API doesn't even exist yet! The next stage is to write the minimum amount of code to get the test *compiling*. That's all, just a clean compile, so you can run the test (which at this stage will fail). IDEs such as IntelliJ IDEA or the open source Eclipse will generate missing classes and implement missing methods for you. Now, and only now, you write the application code to satisfy the test. The final piece of the puzzle is to refactor the code so it's as simple as it can be. This then becomes your development rhythm: write a test, write some code, refactor.

Writing the test before you write the code focuses the mind – and the development process – on delivering only what is absolutely necessary. In the large, this means that the system you develop does exactly what it needs to do and no more. This in turn means that it is easy to modify to make it do more things in the future as they are driven out by more tests.

We keep the tests we wrote and run all of them, often, to make sure the system does everything it is supposed to do (and to alert ourselves immediately if we break any existing functionality). However, the extremely useful test suite we've created is very much a secondary benefit of the TDD process.

So when you're sitting in an interview and someone asks you about test-driven development, remember that it's not about the tests; it's about seeing how little you actually need to do and how cleanly you can do it! If someone asks you to fill a room with oranges? Well, I'll leave that to you.

**Dan North** has been writing software for 12 years, and is a programmer and coach for ThoughtWorks (www.thoughtworks.com), a software development consultancy, where he encourages people to write tests.

*dan.north@thoughtworks.com*

SYS-CON MEDIA

President and CEO:
Fuat Kircaali *fuat@sys-con.com*
Vice President, Business Development:
Grisha Davida *grisha@sys-con.com*
Group Publisher:
Jeremy Geelan *jeremy@sys-con.com*

**Advertising**
Senior Vice President, Sales and Marketing:
Carmen Gonzalez *carmen@sys-con.com*
Vice President, Sales and Marketing:
Miles Silverman *miles@sys-con.com*
Advertising Sales Director:
Robyn Forma *roybn@sys-con.com*
Director, Sales and Marketing:
Megan Ring *megan@sys-con.com*
Advertising Sales Manager:
Alisa Catalano *alisa@sys-con.com*
Associate Sales Managers:
Carrie Gebert *carrieg@sys-con.com*
Kristin Kuhnle *kristen@sys-con.com*

**Editorial**
Executive Editor:
Nancy Valentine *nancy@sys-con.com*
Associate Editors:
Jamie Matusow *jamie@sys-con.com*
Gail Schultz *gail@sys-con.com*
Jean Cassidy *jean@sys-con.com*
Jennifer Van Winckel *jennifer@sys-con.com*
Online Editor:
Lin Goetz *lin@sys-con.com*

**Production**
Production Consultant:
Jim Morgan *jim@sys-con.com*
Lead Designer:
Louis F. Cuffari *louis@sys-con.com*
Art Director:
Alex Botero *alex@sys-con.com*
Associate Art Director:
Richard Silverberg *richards@sys-con.com*
Assistant Art Director:
Tami Beatty *tami@sys-con.com*

**Web Services**
Vice President, Information Systems:
Robert Diamond *robert@sys-con.com*
Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Christopher Croce *chris@sys-con.com*

**Accounting**
Accounts Receivable:
Kerri Von Achen *kerri@sys-con.com*
Financial Analyst:
Joan LaRose *joan@sys-con.com*
Accounts Payable:
Betty White *betty@sys-con.com*

**SYS-CON Events**
President, SYS-CON Events:
Grisha Davida *grisha@sys-con.com*
Conference Manager:
Michael Lynch *mike@sys-con.com*
National Sales Manager:
Sean Raman *raman@sys-con.com*

**Customer Relations**
Circulation Service Coordinators:
Niki Panagopoulos *niki@sys-con.com*
Shelia Dickerson *shelia@sys-con.com*
Edna Earle Russell *edna@sys-con.com*
JDJ Store Manager:
Rachel McGouran *rachel@sys-con.com*

# More little black lines than ever before.

***Web Deployment –**
Implement dynamic web reporting, including queuing and caching of user requests for efficient management of user loads.

**Exports –**
Export in multiple formats: XML, PDF, DHTML, Excel, Word, RTF, text, and email.

***Data Access –**
Access almost any data with native, ODBC, OLE DB connectivity to relational, OLAP, XML, and enterprise data sources.

***Custom Report Style Templates –**
Create and apply a consistent look and feel across multiple reports.

***Report Parts –**
Specify existing key report elements, Report Parts, for dynamic access via wireless or web applications.

**Training and Technical Support –**
Get fast, maximum results with online or onsite training, and a full range of technical support options.

**Extensible Formula Workshop –**
Select from over 300 included functions and operators.

**Java, .NET, and COM SDKs –**
Integrate reporting/presentation layers with virtually any application.

**Report Designer –**
Create highly formatted reports without any programming.

***Unlimited SQL Control –**
Deliver complete control over database connectivity.

***Crystal Repository –**
Store, manage, and reuse report objects.

**Online Developer Zone –**
Use extensive online samples and resources to reduce development cycles, coding, and learning curves.

**Customizable report viewers –**
Use multiple viewer controls including: Java, .NET, DHTML, and ActiveX®.

**Advanced Parameters –**
Deliver dynamic reports with pre-defined parameters for unique data views at runtime.

***new or updated features with Crystal Reports 9 Advanced**

With over 50 new features inside Crystal Reports 9, it's impossible to list all the finer points of our high-productivity reporting solution.

**More features. More productivity. More flexibility.**

Allowing detailed control of end user report viewing and runtime interaction, Crystal Reports is an ideal solution for developers seeking efficient integration of dynamic application reporting and presentation layers using Java, .NET, and COM.

Crystal Reports is also a fundamental component of Crystal Enterprise, a web-based information delivery system. This reliable, scalable, and secure technology is an excellent starting point for application developers who need a platform that can grow.

With over 14 million licenses shipped, and chosen by industry-leading technology partners like Microsoft and SAP for its high-productivity solutions, Crystal Decisions is a trusted software vendor.

Find out more. Visit our web site for free online seminars, the Developer Zone, demos, and evaluation software at: www.crystaldecisions.com/lbl/.

To buy now, call 1-800-877-2340, visit our online store at www.crystaldecisions.com/lbl/estore/, or contact your reseller.

Access. Analyze. Report. Share.®

crystal decisions™

**Joseph Ottinger**
J2EE Editor

# It Just Works

We tend to see the United States through a lens made up of its major population centers: New York; Los Angeles; Washington, DC; Miami; Atlanta; Chicago; and a few others. That's because these are the places that have things "going on," and as a result we get a skewed picture not only of what the United States is about, but of what the United States actually is. From this bird's-eye view, you get the sense that America is all about urban angst, hip-hop, people crammed into shiny metal boxes.

The truth, however, is quite different. Those things are part of America, to be sure, but the wider reality is that America is also open, made up of small, peaceful towns; students going home after school to work on the farm; kids playing pick-up baseball games in open sandlots; shopkeepers talking to their customers by name. America's a true melting pot of color and culture – and because of the lens through which we see her, it's hard to pick out the good among all the bad.

I believe that Java can be looked at the same way, with different groupings based on operating systems, language, aptitude, and application, among other things. By the same measure, we look at our substrata (Java) through a lens the same way a nation is looked at. We see the "high-water marks," the outliers who stand out from the crowd, and as a result it's very easy for us to get a bit misled about where the stream really stands, you might say.

I suspect that Java's doing much better than people fear. There have been lots of events lately (the "Java Desktop," Bill Joy's resignation, and Merrill Lynch's rather pessimistic recommendation to Sun that it spin off Java, among others) that focus on the negative, that aim attention at where Java hasn't succeeded, or, possibly, where it has yet to be successfully leveraged. This is focusing on the choppy surface, ignoring the calm beneath.

I think there's a great chance that past all the hype – positive and negative – Java's doing very well; it's very robust, it's very capable, and the negativity needs to be seen for what it really is – noise and fury aimed at the infrastructure in which Java exists. Are there power plays and egos at stake? Certainly…but do they really affect the ordinary Java developer? I'd say no.

What you're seeing from the industry analysts, academics, and the press is equivalent to Sim City from the city planner's perspective, and I think the Sims themselves are bopping around happily in sometimes unexpected (and unexpectedly successful) ways.

• • •

Lately, I've been working on a set of applications using some very nice hardware and software, things that haven't been getting a lot of press coverage: a Sparc laptop (Tadpole Computer's SparcLE), Solaris 9, the poor old J2SDK 1.4.2, Orion, and a variety of editors including Eclipse, IDEA, and JBuilder 9. Honestly, I've surprised myself: it's been an eye-opening experience, reminding me of how nice all this is. These things aren't sexy anymore, as far as I can see. They just work, and work well. I'm not fiddling about with cutting-edge stuff, hoping that it'll come together in time to create a successful application; I'm using the standards to do what I need them to do, as building blocks for an application that does what I need. Perhaps these things don't have the bullet-list success that others do, and I'm fine with that – I'd rather just get things done. This stuff isn't rocket science unless we make it that way, and we don't have to. One of Java's strengths is in abstraction, so that we don't have to be the academic upper crust, working with obscure technical epistemology to obviate technical detritus to accomplish minutiae; we just put things together so they work. Sure, we may not always tune things specifically for a given platform or solution space – but we can do what we need faster, with fewer bugs, and with more portability than anyone else.

This stuff rocks.

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

*josephottinger@sys-con.com*

# Managing J2EE Systems with JMX and JUnit

## Manage new systems as they are developed

Lucas McGregor

The promise of J2EE was to build more robust, scalable, and secure enterprise systems. J2EE promised that we could do it quickly and easily since J2EE is supposed to take the complexity out of building powerful distributed systems. But as with the J2EE spec itself, these systems usually suffer through management only as an afterthought.

Many management systems focus on proprietary interfaces that react to specific events. They offer solutions in which your management is tied up with the system you are managing. JMX4ODP decouples testing and management from the target system and focuses on using reusable components that are bound and deployed using XML configurations.

This article walks you through the process of setting up a basic service monitor and event handler for a common J2EE *n*-tier system. Developers of J2EE systems will be able to use JMX4ODP to create testing suites to help them develop more reliable systems. J2EE application administrators will be able to use JMX4ODP to simplify and regulate the management of deployed systems.

### Technologies Used

JMX4ODP is built from open standards technologies because they offer a better return on their investment by leveraging the work and experience of the community. Open standards also leverage your work and experience; for example, if your project has been tested in the past, you probably have someone on staff who already has JUnit experience. If you don't, learning JUnit will be useful for future projects.

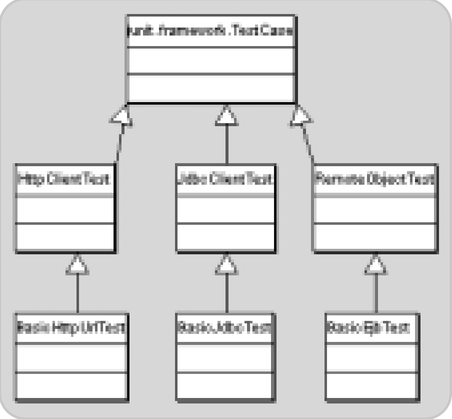


**Figure 1** JUnit service testers

JUnit and JMX are the two core foundations for JMX4ODP's approach to management. JUnit is becoming the de facto standard for unit testing. JUnit support is common in most IDE and test suites. JMX is Java's official answer to system management. Groups like JBoss are pushing JMX even further to make it a key piece in building complex infrastructures. JMX support is common in major J2EE application servers. IBM has even integrated JMX functionality with its popular Tivoli suite.

### What You Need

To implement the examples in this article, make sure you have the following libraries available:

- ***JMX4ODP:*** The core diagnostic and event manager suite (http://jmx-4odp. sourceforge.net).
- ***JUnit:*** The de facto standard for unit testing of Java components (www.junit. org).
- ***JMX:*** The J2EE Java Management Extensions (www.javasoft.com/jmx).
- ***JMX Remoting:*** The JSR to expand JMX to include remote access (http://jcp.org/en/jsr/detail?id=160).
- ***Xerces:*** The popular Apache XML parser (http://xml.apache.org/ xerces -j/index.html). *Note:* JMX4ODP has not been ported to use Xerces 2 yet, so you'll need Xerces 1, which ships with the current JBoss, WebLogic, and WebSphere.
- ***JavaMail:*** The J2EE extension for e-mail (http://java.sun.com/ products/javamail/).

### JUnit as a Diagnostics Tool

JUnit bills itself as a regression-testing suite for code objects, but it's not much of a leap to see it as a tool for distributed system diagnostics. JUnit runs tests by instantiating objects, invoking their methods with known inputs, and checking the output against expected returns.

Distributed systems are built over time as a collection of services – some standardized, some proprietary. Each service can be treated as an object for JUnit to test. In a typical J2EE installation, you have HTTP daemons, servlet engines, JNDI trees, RMI-enabled EJB containers, and databases accessed via JDBC.

Figure 1 illustrates these services as extensions of common protocols or as classes of objects to test. Since a system has a limited number of supported protocols, you can save a lot of coding by creating a base test class for each protocol and extending it as needed. The JMX4ODP's org.jmx4odp.junit DiagnosticWorkers package contains classes for testing HTTP, RMI, and JDBC services.

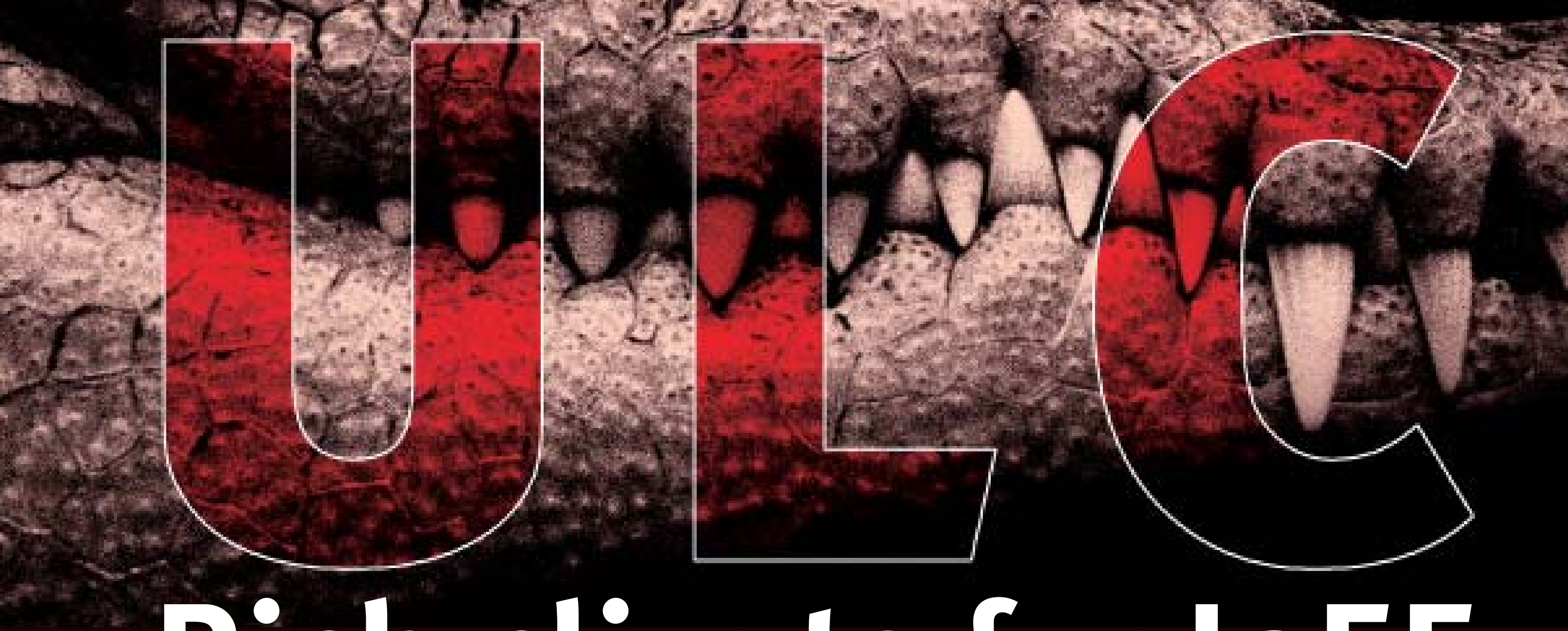
### JUnit Test for Services

The JMX4ODP test classes follow the JUnit "assert" pattern. Each possible test method name starts with "assert," allowing developers to easily identify testing methods versus utility methods. Each method is stateless, allowing multiple testing objects to utilize the same underlying protocol test object, e.g., the HttpClientTest object contains methods for acquiring an HttpURLConnection and testing the connection for HTTP statuses and content.

By extending these basic service test classes to encapsulate a series of stateful tests, we get objects that are simply beans that contain a set of tests to run on a service. For example, you could extend the HTTP protocol test class to create an object that checks if you can reach a URL. You could then extend this class to make a test that checks a secured URL.

JMX4ODP's org.jmx4odp.junitDiagnosticWorkers package contains tests for three basic services: HTTP, RMI, and JDBC. Each is implemented as a stateful bean that you instantiate, set parameters for, and then hand over to JUnit to run.

To test HTTP services with BasicHttpUrlTest, set the URL you want

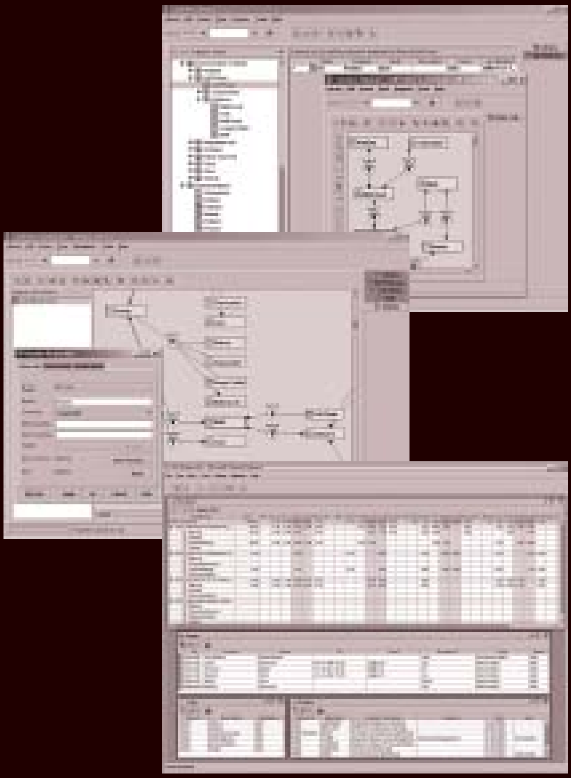


# Rich clients for J2EE

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

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

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



***Benefits for:***

## Development & Maintenance

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

## Deployment & Operation

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

Ready for IBM WebSphere. Studio software

canoo.com

Canoo Engineering AG http://www.canoo.com/ulc/

**Download your free trial today!**

to check and hand the object to JUnit, which will invoke each method that starts with the word “test.” BasicHttpUrlTest’s only test method is “test_URLOk,” which checks the URL for a returned HTTP: 200 OK.

BasicEjbTest tests EJB services. It can use the settings in your jndi.properties to connect to your JNDI tree, or you can set them programmatically. You must set the JNDI name of the EJB. It contains one test, “test_AccessEJB,” which tries to retrieve a RemoteObject from the JNDI tree by the set name and invoke getEJBMetaData upon it.

The BasicJDBCTest is more complicated. You need to set the JDBC URL, a test SQL select statement, and the database username and password. You can set an optional record threshold, which defaults to 1. This test object checks for two things before giving the service a green light. First, it runs “test_CanConnect” to check if JDBC can connect to the database. Second, it runs “test_SelectGood” to ensure that the test SQL select statement returns at least as many records as the threshold is set to.

## Building Your TestSuite

JUnit has the ability to hierarchically arrange tests using TestSuite objects, but TestSuites are maintained programmatically, which is a big maintenance hit. JMX4ODP uses the org.jmx4odp.junitDiagnosticWorkers.SuiteAssembler object to translate an XML file into a TestSuite object, which cuts all the coding from maintenance.

Figure 2 shows the TestSuite XML entities used by the SuiteAssembler. TestSuite objects can hold either another TestSuite or TestCase. A TestCase is one of the stateful test beans. You invoke the get/set methods for service and test properties such as URLs and thresholds with the INVOKE element, which can take arguments of java.lang.String, boolean, and int. A TestSuitexml to test Yahoo’s Web servers would look like:

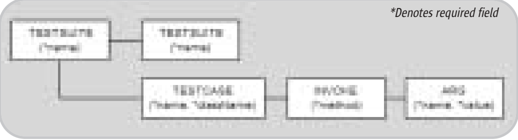


**Figure 2** TestSuite.xml document objects used by SuiteAssembler

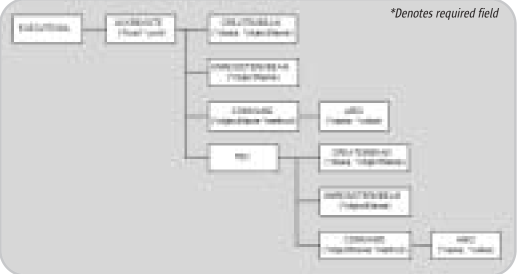


**Figure 3** Executive.xml document objects used by XMLExecutor

```
<TESTSUITE name="Web Servers" >
    <TESTCASE name="Yahoo" className=\
   "org.jmx4odp.junitDiagnosticWorkers\
   BasicHttpUrlTest" >
       <INVOKE method="setUrl" >
           <ARG type="java.lang.String"
           value="http://www.yahoo.com"
           />
       </INVOKE>
   </TESTSUITE>
```

SuiteAssembler will parse the XML and generate a TestSuite called “Web Servers” that contains a single BasicHttpUrlTest test bean called “Yahoo”. It then hands the TestSuite over to JUnit, which will run the BasicHttpUrlTest.test_UrlOk method to see if a connection to www.yahoo.com returns an HTTP 200:OK. If Yahoo is unreachable or returns a different status, JUnit will display a failure.

You can use JUnit’s junit.swingui .TestRunner to run JUnit tests in a graphical interface. Pass TestRunner the name of your test class as an argument. The JMX4ODP SuiteAssembler will load a TestSuite.xml file in the working directory. If you use the example TestSuite.xml and type:

```
> java junit.swingui.TestRunner
org.jmx4odp.junitDiagnosticWorkers.SuiteAsse
mbler
```

in the same directory, JUnit will parse the file, try to connect to www .yahoo.com, and display the results.

At this point you could simply build a TestSuite.xml file to check all the services you want and just use JUnit to run on-demand diagnostics of all your systems. If the line is green, the servers are clean. This would certainly be handy at 3 a.m. when you get the “the site is acting weird” phone call.

Already you can quickly and repeatedly run a set of known tests on your site and identify problems. However, this is a reactive instead of proactive solution. We now need to automate these tests and feed the results to a system that can use them.

## Using Tests to Manage with JMX

Before JMX, there was no Java standard way for starting, stopping, monitoring, and managing components. If you’re not familiar with JMX, there are some great books, such as *JMX: Managing J2EE with Java Management Extensions* by Marc Fleury, Juha Lindfors, and The JBoss Group.

JMX is a powerful and convenient way of building loosely coupled systems. The JMX agent is a bean container for specialized management beans called MBeans. The agent allows you to instantiate new MBeans, register existing MBeans, bind MBeans together, and send and receive notifications.

Many J2EE engines and management packages have adopted JMX as a core feature, because it’s flexible and extensible. J2EE programmers are familiar with component-based programming, and JMX capitalizes on that to create component-based management systems that are scalable. JSR 160 (www.jcp.org/en/jsr/detail? id=160) is extending the core JMX specification to include remoting functionality, which is used by JMX4ODP to connect clients and other agents to each other via RMI.

### *EventRunner*

org.jmx4odp.junitRunner.Event Runner is an MBean that implements a JUnit TestRunner. It serves as the bridge between your JUnit diagnostic setup and JMX management. It will fetch a TestSuitexml from a given URL, use the SuiteAssembler to construct a TestSuite, hand it to a JUnit TestRunner to run all the tests, and broadcast any failures or errors as notifications to any registered listeners. The EventRunner will then sleep for the specified time and do it all over again.

You can think of JMX4ODP as a reflex system for your J2EE system. JUnit test objects act as live nerves gathering information about the state of objects and services in your system. The JMX agent is like a spinal cord, transmitting these impulses between the brain and muscles. The EventRunner MBean, the system’s brain, coordinates all the JUnit tests and keeps them running regularly. Now you just need to add some muscle to complete the system.

Muscle takes the form of JMX NotificationListener MBeans. Any MBean that implements the javax. management.NotificationListener interface can register itself with the EventRunner to receive failure and error notifications. The NotificationListener will receive a javax. management.Notification object that contains attributes including timestamp, type, and message. The type of message can be set via the two EventRunner methods: setErrorTopic(String s) and setFailureTopic(String s). The event runner will broadcast the message as

an error type if an error occurred while trying to run the test. It will use the failure topic if the test was unsuccessful. The event notification message will be formatted like TestFailure.getName() + ": " + failure.toString(); if failure.toString() returns a null, it will use failure.thrownException().toString().

A NotificationListener can register with the EventRunner and be activated upon these events. It can use the Notification.getMessage() to learn which test failed and how. We'll use the org.jmx4odp.notificationWorkers.NotificationMailer as a simple starting place. This MBean uses the javax.mail.* package to e-mail JMX notifications. By registering the Notification Mailer with the EventRunner, you have a failure notification system that will alert your sysadmin when a problem occurs.

All this needs to be set up programmatically which is a maintenance issue. JMX's MLet object can be used to make a JMX agent load MBeans specified in an XML file, but it doesn't include the ability to invoke functions on instantiated MBeans. To overcome this, JMX4ODP uses its org.jmx4odp. j4oNet.XmlExecutor object to load and access MBeans in an agent.

Figure 3 shows the XML entities used by the XmlExecutor. Listing 1 shows the beginning of such a file.

Everything is a child of the EXECUTEXML element. The JMXREMOTE element tells XmlExecutor to connect to the JMX agent's RMIAdaptor. TRY groups elements together, so if one fails, it will skip the rest in the group so you don't have to wait for each element to fail. Use CREATEMBEAN to tell the JMX agent to instantiate a new MBean. COMMAND is the big advantage of XmlExecutor; it allows you to invoke methods on existing MBeans.

### *BaseAgent*

JMX4ODP ships with org.jmx4odp. j4oNet.BaseServer as its JMX agent. It takes the HTMLAdaptor port and the RMIAdaptor port as its two arguments. Type

```
> java org.jmx4odp.j4oNet.BaseServer 8080
1099
```

to start the BaseServer. Now point a Web browser to localhost:8080 to see an HTML interface of your MBeans. JMXREMOTE will use port 1099.

Now that you have a JMX agent with an RMIAdaptor running, you can create an XML Execute.xml file that will tell XmlExecutor to:

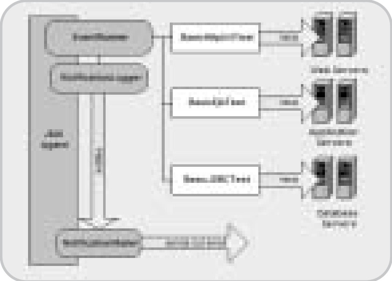


**Figure 4** JMX/JUnit event notification system

1. Create an EventRunnerMBean.
2. Invoke EventRunner.setFailureTopic to set the failure Notification type.
3. Invoke EventRunner.setErrorTopic to set the error Notification type.
4. Invoke EventRunner.SetSuite-AssemblerConfig to give the URL for your TestSuite.xml.
5. Invoke EventRunner.setSleepCount to set how many milliseconds to sleep between test cycles.
6. Create a NotificationMailer MBean.
7. Invoke NotificationMailer.setSmtp-Host to set the host name of your mail gateway.
8. Invoke NotificationMailer.setSmtp-User and setSmtpPassword to set the username and password for your SMTP user if needed.
9. Invoke NotificationMailer.setSmtp-Port if your gateway uses anything other than port 25.
10. Invoke NotificationMailer.setFrom Address to the address you want the notification e-mails to come from.
11. Invoke NotificationMailer.setSubject to set the e-mail subject line.
12. Invoke NotificationMailer.addTo Address to add an address to which to send notification e-mails.
13. Invoke NotificationMailer.setActive to activate the mailer; otherwise it will ignore all notifications while inactive.
14. Invoke NotificationMailer.add ListenedToObject to ."MONITOR: name=EventRunner,Notification-Logger=true," which will tell Notification Mailer to listen to the Notification Logger created by the EventRunner.
15. Invoke EventRunner.startDaemon to start the testing cycle.

Figure 4 illustrates how JMX4ODP combines the JUnit tests and JMX management components to create a management system. If you use the example TestSuite.xml, you're testing if you can reach Yahoo. If this test fails, a JUnit event will be broadcast to all registered listeners. Your only listener right now is an MBean that will send out the event as an e-mail. If you start up a Web browser to http://localhost:8080, you'll see your JMX agent's HTMLAdaptor and all the new MBeans you started.

## We've Only Just Begun...

The most obvious ways to expand JMX4ODP is to:

- Create custom tests.
- Create custom NotificationListeners.

The example we just used showed how to create an alert system. However, you could easily create a NotificationListener MBean that alters the system's state instead of telling your sysadmin to do it. For example, if your JBDC tester finds that your primary database has gone down, you could have a NotificationListener automatically start the failover procedure. Other possible directions would include automating start-up and shut-down via custom NotificationListeners; or monitoring log file size, network traffic, or response time via custom JUnit test beans.

JMX and JUnit allow for loosely coupled designs that are modular, making it trivial to add and rearrange tests and responses. To add new actions, register new NotificationListeners. To change your test plans, update an XML file. Maintenance of your management suite should not be a full-time job. Your diagnostics and testing systems are no longer a coupled part of the target system. Once you've created common components, they can be used to manage new systems as they're developed.

By using the JMX4ODP mode of design, management isn't an afterthought, but neither is it a burden of intense design or the lock-in of proprietary solutions. Management becomes another one of those things you get free when you use J2EE.

## Resource

- *JMX:* http://java.sun.com/products/JavaManagement/index.html

**Listing 1** Code template for a custom task

```
<EXECUTEXML>
   <JMXREMOTE host="localhost" \
   port="1099">
   <TRY>
        <CREATEMBEAN
class="org.jmx4odp.junitRunner.\
        EventRunner" objectName=\
        "MONITOR:name=EventRunner"
/>

         <COMMAND objectName=\
         "MONITOR:name=EventRunner"
         method="setErrorTopic">
            <ARG
type="java.lang.String"
            value="error" />
         </COMMAND>

...
```

**Lucas McGregor** is the CTO of Xdrive, where he manages their architecture and technology. Over the years he has designed distributed management systems for tier-one national ISPs, massive online gaming systems, and consulted-on projects ranging from distributed intelligent agents to automated protein analysis. He is interested in the architecture of robust and manageable distributed systems.

*mcgregor_lucas@hotmail.com*

# Enabling a File System as a Transactional Resource

## Building an adapter

by Ashish Garg and Sandip Mane

Transactional support is fundamental to application development. While most data sources are transactional in nature, some data sources like the file system are not.

In typical J2EE deployments, applications often need to interface with other applications through file-based messages. Such deployments would benefit from an adapter that would buffer all file operations and provide a transactional view to the file system.

This article, targeted at developers, demonstrates how J2EE Connector Architecture and JTA specifications can be implemented to build such an adapter, XAFileConnector. We'll also suggest enhancements to extend the adapter functionality in light of the new proposed draft connector specifications.

Most applications are bound to the local file system. They use the file system to store and share data. Often the success of interactions with the file system needs to be tied to the successful completion of other actions involving other Enterprise Information Systems (EIS). For instance, we often encounter situations where a record is to be deleted from the database only after it has been successfully written to a file. While the same can be easily built into the application logic, it would be infinitely more neater if we could develop an extension on the file system that would help us include the file operations such as "create", "read", "update", and "delete" as part of a transactional unit of work that could be committed or rolled back as a group. This extension is called a Resource Adapter. The ubiquitous presence of the file system and the role it plays in Enterprise Application Integration strategy justify the need to build such a resource adapter.

For a J2EE-managed environment, i.e., an environment characterized by the presence of a container for runtime support, J2EE recommends Connector Architecture as a standard way of integrating heterogeneous EISs with the application server. The Connector Architecture specifies application- and system-level contracts to be implemented by the EIS. The resource adapter implements the EIS side of the contracts. The adapter uses a native interface specific to the underlying EIS, runs in the containers address space, and manages access to the EIS resources. Adherence to the Connector contracts on one hand ensures that the same adapter can be deployed on different vendors' containers, while ensuring that application and tool developers have a standard interface to program to for accessing different EIS systems. Among the set of contracts that need to be implemented by the resource adapter to seamlessly plug-in to any J2EE platform is the XAResource transaction contract. This contract defines the communication interfaces between the parties involved in a distributed transaction. Implementation of this interface by the adapter enables seamless propagation of transaction context and allows the associated resource manager (RM) to participate in a two-phase commit protocol along with other resources.

With this introduction we proceed to discuss what these contracts are and what it takes to implement these contracts to build an adapter.

### Understanding and Implementing the Contracts

The application contract manifests itself as a Common Client Interface (CCI). CCI is the client side of the adapter and performs the following roles:

- Specifies how application components connect to the EIS through a resource adapter
- Provides an API for coding EIS function calls and retrieving results

#### *Connecting to the EIS*

ConnectionFactory is the application's gateway to the EIS. It's instantiated at the time of deployment by the container and a reference to it, or its serialized representation, is bound to the JNDI namespace. Later the application components do a JNDI lookup to get hold of the factory to create connection objects to the EIS. The ConnectionFactory does not represent the actual connection repository, nor is the connection the actual physical connection to the EIS. Both are client-side proxies that nevertheless maintain handles to the actual factory and connection objects, respectively. The container plays the role of an intermediary between the real objects and their proxies, and the mechanism allows the container to inject value-added services by intercepting calls made on the proxies before they're actually delegated to the real objects.

When a ConnectionFactory encounters a getConnection request, it delegates it to the ConnectionManager instance.

```
public Connection getConnection()
                    throws ResourceException {

return (Connection)connMgr.allocateConnection
    (mngdConnFactory, new ConnectionRequestInfoImpl());
}
```

In a managed environment the ConnectionManager implementation is provided by the container. The ConnectionManager maintains a pool of connections corresponding to each factory. The ConnectionManager checks whether it can service the request from the pool. If not, it requests the real factory (ManagedConnectionFactory) to create a new connection.

How does the container know about the availability status of a connection?

The Connector Architecture prescribes an elaborate mechanism of listeners to implement callbacks. Whenever a new connection is created, the container registers a listener with it. The connection raises event notifications to intimate the container about the happenings on the connection. When a close is called on the connection proxy it terminates its association with the physical connection. The physical connection in turn generates an event. All the listeners registered on the connection can react to the event. The container on its part uses the event to change the availability status of the connection from in-use to available, i.e., if the connection is not participating in a transaction. If the connection is participating in a transaction, the container waits for the transaction to commit before it can make the connection available (see Listing 1).

### *Invoking EIS Functions*

An InteractionSpec implementation holds properties for driving an interaction with an EIS instance. In our case each property maps to a File Operation that may be performed.

```
public static final int CREATE = 10;
public static final int UPDATE = 20;
public static final int DELETE = 30;
public static final int MOVE   = 40;
public static final int SKIP   = 50;
public static final int READ   = 60;
```

Since CCI is EIS independent it cannot use any of the EIS data structures. To get over this, CCI introduces the concept of a record. A record is a generic representation of data that's exchanged with the EIS. More specific implementations to represent hierarchical, tabular data collections can also be implemented. XAFileConnector extends the MappedRecord, which is a key value representation of record elements for both input and output records (see Listing 2).

CCI defines an interaction interface that allows a client to interact with EIS. An interaction represents a single communication with the EIS, an EIS function call. An interaction instance is obtained from a connection. The interaction instance is required to maintain its association with the connection instance. The interaction delegates its execution to the connection, which in turn delegates its execution to the associated ManagedConnection (see Listing 3).

## Transaction Contract

A transaction as we know it is a set of operations performed in such a way that the state of the system after the transaction is either the cumulative effect of state changes due to successful individual operations or is the initial state before the commencement of the transaction in case one or more operations happens to fail. Transactions can be classified into two types based on the number of resources participating.

1. A local transaction performs operations on a single RM. The local transactions can be further classified based on the manner of demarcating transaction boundaries.
   - ***javax.resource.cci.LocalTransaction:*** These run under the control of the RM. The transaction object can be obtained from the connection object and used to demarcate transaction boundaries somewhat analogous to the way JDBC setAutoCommit(false) and commit APIs are used to demarcate transaction boundaries.
   - ***javax.resource.spi.LocalTransaction:*** Work in this type of transaction can accrue over multiple client components. A client component may do some work on a resource and pass control to another component, which may again do some work while still being part of the same transaction. The transaction context is passed along with the control, and it contains the following information:
     – A reference to the transaction manager (TM), the external entity that is used to demarcate transaction boundaries
     – A globally unique transaction identifier, XID, generated by the TM
     – A transaction timeout time
2. Global/distributed transactions – work in this type of transaction can span multiple RMs. The transaction manager demarcates the transaction boundaries. A TM coordinates the activities on the participating RMs using the XAResource interface. An RM knows only about the work it does for its transaction branch. The XAResource interface implements the two-phase commit protocol between the RMs and the TM. The XAResource interface allows for a one-phase optimization in case only one resource is participating.

With this brief introduction to transactions we'll start with how we can incorporate transactional ability into our file system adapter.

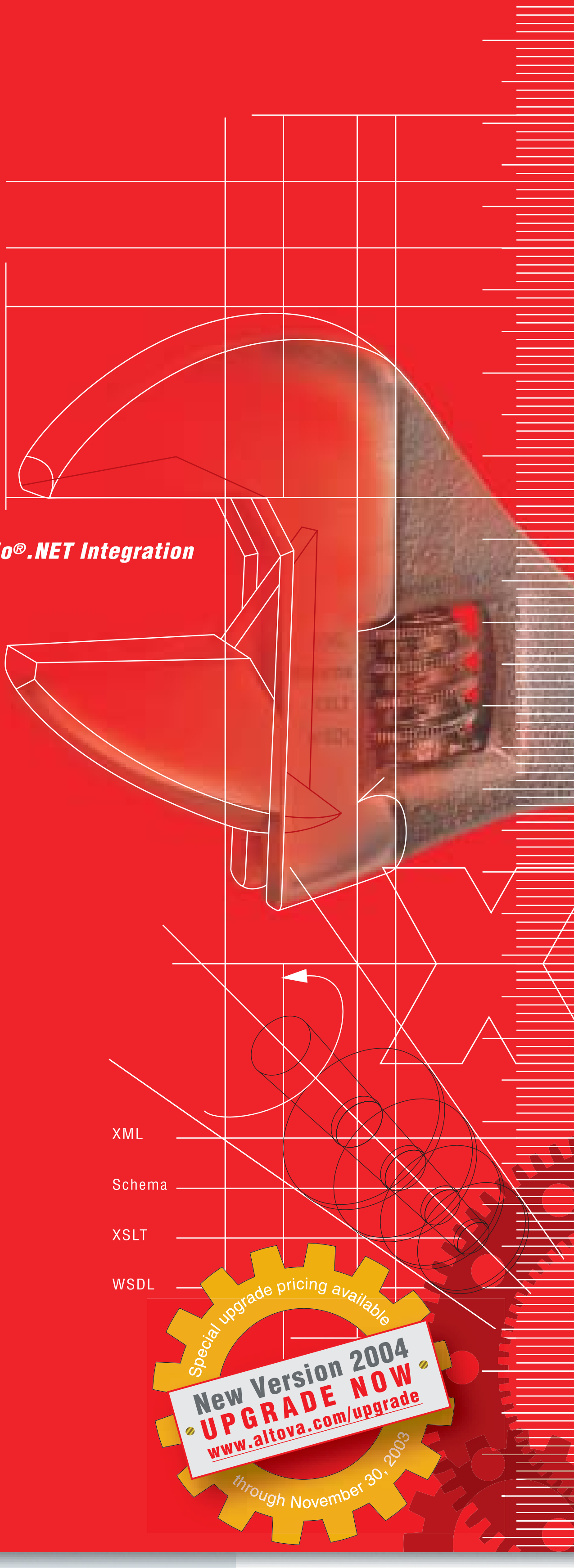
For a file system adapter to be able to roll back a transaction, i.e., revert to state prior to the start of a transaction, an adapter must either memorize the original state or it must be able to get back to the original state by performing certain anti-operations that reverse the effect of the operations. To ensure feasibility and consistency at all times, the anti-operations have to be performed in the order that is exactly the reverse order in which the operations were performed. Similarly, committing a transaction should flush all transactional logs to free all memory.

For each file operation on the connection that modifies the state of the file system (nonread only call), XAFileConnector adds a WorkItem to the associated transaction's work list. If there's no active transaction associated with the connection, it autocommits the work and no WorkItem is created. The WorkItem stores enough information about the associated operation to be able to reverse its effect in case of a rollback or to do a cleanup of the backup in case of a commit. A work area is designated to store the information needed for restoration in the event of a rollback (see Listing 4).

Note how MOVE and CREATE file operation do not have any associated cleanup during operation commit while UPDATE and DELETE operations delete the backups.

Depending on the kind of transaction running, javax.resource.cci.LocalTransaction, javax.resource.spi.LocalTransaction, or XAResource keeps track of the work being done in the transaction impending completion.

CCI Local Transaction implementations are obtained directly from the connection, without the container having any role to play. Since the container is not directly involved, it is not intimated of the state of the transaction, so CCI implementations of local transactions differ from SPI implementations in the sense that they need to raise event notifications to apprise the container of transaction life-cycle events. The events help the container in connection pool management. Since there's only a marginal difference in functionality, the same class is used to implement both forms of local transactions with a flag identifying the type of transaction the current instance represents.


XML
Schema
XSLT
WSDL
Special upgrade pricing available
New Version 2004
UPGRADE NOW
www.altova.com/upgrade
through November 30, 2003

Our implementation model exhibits the following relationships shown in Figure 1.

Each ManagedConnection can provide an XAResource implementation for distributed transaction management. The beginTransaction call associates the application component's thread of control with a global transaction. The TM generates a globally unique XID and calls start on all open RMs that are linked with the thread. The RMs are enlisted with the transaction. To guarantee scalability of the Connector Architecture, it recommends that a physical connection and hence the associated XAResource can participate in multiple transactions, but at any point only one of these transactions can be active. To ensure this, the start call first checks that the associated connection isn't running a local transaction nor is the XAResource instance associated with an active transaction. Once assured, it associates the passed XID with the XAResource instance and activates the transaction. All work done henceforth on this physical connection, until the transaction is suspended or completed, accumulates against the active transaction.

The start call is accompanied by the following flags:

- ***TMNOFLAGS:*** Indicates that it is a new transaction. A call to getTransactionState with flag true creates new entries for the new transaction and initializes the instance.
- ***TMRESUME:*** Indicates a suspended transaction is being resumed. A call to getTransactionState with flag false means no new entries need to be made. The state of the transaction is retrieved and the current instance is initialized with that state. The state of the transaction is also changed from SUSPENDED to NOT SUSPENDED.
- ***TMJOIN:*** A two-phase optimization, the TM, when it detects that a particular RM instance is already participating in a transaction, instead of creating a new branch decides to merge all work done on a single RM against a single XID. By doing this it saves on running the two-phase protocol on all branches separately. No new entries have to be made and the state of the transaction remains throughout NOT SUSPENDED (see Listing 5).

The application component calls commit to make the effects of the transaction permanent. The TM first calls end for each involved RM, from the AP's thread of control, to dissociate the thread from the global transaction. The TM then executes the two-phase commit protocol.

STAGE 1 is the prepare stage or the voting stage. TM calls prepare for each RM that was associated with the global transaction. RMs express their readiness to commit the transaction. A single negative vote ensures rollback. A prepare attempt on a suspended transaction would throw back a protocol exception. Similarly it throws back an exception, albeit a different one, when the transaction has been marked earlier for rollback due to internal errors. Prepare stage also offers a two-phase optimization. A transaction that has not changed the state of the RM doesn't need to go through the second phase if prepare returns XA_RDONLY (see Listing 6).

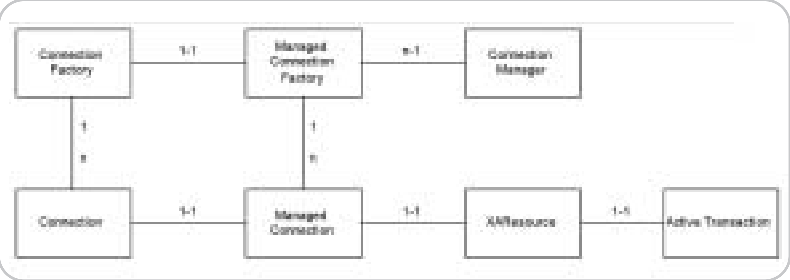


**Figure 1** Implementation Model

Stage 2 is the commit stage. If all RMs return success from prepare, the TM records a decision to commit the transaction and calls commit for each RM. The XA specification allows for an optimization in this procedure. If the TM has dealt with only one subordinate RM in the global transaction, it can omit Phase 1 by directly calling commit with the onePhase flag set to true (see Listing 7).

## Deployment Process

Before we can start using the adapter, we need to deploy it. All adapter interfaces, classes, native libraries, etc., along with the Deployment Descriptor (DD), are packaged in a .rar archive. In situations where multiple J2EE applications need to share the XAFileConnector, it can be deployed directly into an application server as a standalone unit. However, the resource adapter module may also be bundled with a J2EE application, if it is needed only by a single application.

The Deployment Descriptor provides a lot of information to the container, including:

- General information such as name and version of adapter, vendor name, licensing requirements, etc.
- Names of concrete classes for certain core interfaces; for example, we have to provide the name of the class that implements the ManagedConnectionFactory Interface.
- Information about the kind of support built into the adapter. This allows the container to perform certain optimizations. XAFileConnector offers full support for XATransaction. No authentication support is needed as all native calls to connect to the resource, i.e., the local file system, are made in the current user's context.
- Configurable properties that are dependent on the deployment environment. The property naming conventions follow the JavaBean standard. XAFileConnector uses a property by the name of workingFolder to store the name of the folder where transaction logs of the running transactions are kept. The deployer sets this property at the time of deployment by invoking the corresponding setter method. The method checks for the existence of the folder and sufficient access permissions before returning (see Listings 8 and 9).

To be able to successfully deploy an adapter, a better understanding of the deployment process is essential. We will briefly walk through the responsibilities of the deployment code and how it interacts with the adapter code.

- The deployment code looks at the DD to find out which class implements the ManagedConnectionFactory Interface. It dynamically creates an instance of the same and configures properties on the instance.XAFile The Connector uses only one property workingFolder as discussed earlier.
- In a managed environment the deployment code instantiates a ConnectionManager specific to the container and obtains a ConnectionFactory instance from ManagedConnectionFactory, as shown in the following implementation:

```
public Object createConnectionFactory(ConnectionManager cxManager)
throws ResourceException
{
return new ConnectionFactoryImpl(this, cxManager);
}
```

Note how the ConnectionManager instance is passed around while creating the ConnectionFactory. The mechanism helps associate the two factories and the ConnectionManager. The ConnectionManager is the ConnectionFactory's interface to the container.

# Get the complete picture

## Features, Performance and Control

### Discover the ILOG JViews Graphics Components

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

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

Quickly and easily build:

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

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

ILOG®

Changing the rules of business™

J2ME | J2SE | J2EE | Home

- The deployment code then binds the ConnectionFactory instance in the JNDI namespace from where it's later looked up by the application component.

## Writing a Client

Clients needing to interact with the file system will have to use the CCI interface. They cannot use the java.io package to access the file system if they want the transactional support. While the approach may look nonintuitive and tiresome, which in fact it is, it is not as big a pain as you would imagine it to be.

By this time we would have understood how to acquire a connection to the file system.

- Once connected we can obtain an interaction from the connection.

```
Interaction interaction = conn.createInteraction();
```

Interaction maintains an association with the connection from which it was created and executes on the same connection.

- A connection object also serves as the source for RecordFactory. RecordFactory hands out all types of records. Records are the means of exchanging data, both incoming and outgoing, with the files system.

```
RecordFactory rf = cf.getRecordFactory();
MappedRecord in = rf.createMappedRecord("IN");
MappedRecord out = rf.createMappedRecord("OUT");
```

- Assuming that we wish to create a file, the only parameters we would need to pass are the folder name in which to create the file and the file name. Initialize the records with the input parameters.

```
in.put("SOURCE_DIR", "E:\\bea\\weblogic700\\FILES1");
in.put("SOURCE_FILENAME", "create.txt");
```

- Now the only information that needs to be passed is our intention of actually creating a file. This is specified when we instantiate an object of the class InteractionSpecImpl.

```
interaction.execute(new
InteractionSpecImpl(InteractionSpecImpl.CREATE), in, out);
```

- Handle the exceptions and examine the OUT record for returned values.
- Close all open resources.

## What Further?

The proposed draft connector specification makes provisions for inbound communication through a Message Inflow Contract and also includes a Work Management Contract. The inclusion of the two would make it feasible for the adapter to work as a poller.

The Work Management Contract would allow the adapter to monitor folders for incoming files. The activity can be performed by submitting work to the container. The contract would save the adapter the task of creating its own threads thus allowing the container to exercise better control over its runtime environment.

The Message Inflow Contract would allow the adapter to trigger action in the event of receiving a file. The situation can be thought of as analogous to a JMS situation, with the adapter instead of an MDB delivering messages by polling on a folder instead of a queue.

Better concurrency control can also be built into the adapter using the new improved I/O support found in newer Java runtimes, 1.4 onwards.

## References

- *JTA specifications:* http://java.sun.com/products/jta
- *JCA specifications:* http://java.sun.com/j2ee/ 'connector/download.html

**Listing 1**

```
public void close() throws ResourceException
{
if (mConn != null)
{
     mConn.sendEvent(ConnectionEvent.CONNECTION_CLOSED,
null, this);
     if (localTransaction != null &&
localTransaction.inTransaction) {
            localTransaction.rollback();
            localTransaction = null;
     }
     mConn = null;
}
return;
}
```

**Listing 2**

```
public class FSInRecord extends HashMap
                 implements MappedRecord
{
   static final String SOURCE_DIR
       = new String("SOURCE_DIR");
   static final String SOURCE_FILENAME
       = new String("SOURCE_FILENAME");
   static final String DESTINATION_DIR
       = new String("DESTINATION_DIR");
   static final String DESTINATION_FILENAME
       = new String("DESTINATION_FILENAME");
   static final String DATA
       = new String("DATA");
   static final String SIZE
       = new String("SIZE");

   public FSInRecord(String fileName,
       String newName, String sourceDir,
       String destinationDir, int size)
   {
      this.put(SOURCE_DIR, sourceDir);
      this.put(SOURCE_FILENAME, fileName);
      this.put(DESTINATION_DIR, destinationDir);
      this.put(DESTINATION_FILENAME, newName);
      this.put(SIZE, new Integer(size));
   }
...
...
}
```

**Listing 3**

```
boolean execute(InteractionSpec ispec, Record input,
Record output) throws ResourceException
{
 FSOutRecord out = (FSOutRecord)output;
 FSInRecord in = (FSInRecord)input;

 String sourceDir = (String)in.get(FSInRecord.SOURCE_DIR);
  sourceDir = (sourceDir != null &&
sourceDir.endsWith(File.separator)) ? sourceDir :
sourceDir + File.separator;
 String sourceFileName =
(String)in.get(FSInRecord.SOURCE_FILENAME);
 String destDir =
(String)in.get(FSInRecord.DESTINATION_DIR);
  destDir = (destDir != null && destDir.endsWith(File.sep-
arator)) ? destDir : destDir + File.separator;
 String destFileName = (String)in.get(FSInRecord.DESTINA-
TION_FILENAME);
 String destFile;

 if((sourceDir == null) || (sourceDir.equals("")))
  throw new ResourceException("Invalid File Directory");
 if((sourceFileName == null) ||
(sourceFileName.equals("")))
  throw new ResourceException("Invalid File Name");
 String sourceFile = new String(sourceDir +
sourceFileName);

 if(sourceDir.equals(workingFolder))
```

**Ashish Garg** is a senior technical specialist with Infosys Technologies Ltd. He has more than four years of experience in J2EE technologies.

*ashish_garg@infosys.com*

**Sandip Mane** is a senior technical specialist with Infosys Technologies Ltd. and specializes in WebLogic.

*msandip@infosys.com*

```
  throw new ResourceException("Invalid File Directory. It
can not be same as working folder directory.");

 try
 {
  switch(((InteractionSpecImpl)ispec).getAction())
  {
   case InteractionSpecImpl.CREATE: {

    File file = new File(sourceFile);
    boolean success = file.createNewFile();
    if(!success) throw new ResourceException("File could
not be created");

    addWork(sourceDir, sourceFileName, null, null,
WorkItem.CREATE);
    break;}

   case InteractionSpecImpl.DELETE: {
    destFileName = sourceFileName + new
java.util.Date().getTime();
    destFile = new String(workingFolder + destFileName);
    closeFile(sourceFile);

    File destFileHandle = new File(destFile);
    File sourceFileHandle = new File(sourceFile);
    boolean status =
sourceFileHandle.renameTo(destFileHandle);
    if(! status)
     throw new ResourceException("File can not be delet-
ed");
    addWork(sourceDir, sourceFileName, workingFolder,
destFileName, WorkItem.DELETE);

    break;}

   case InteractionSpecImpl.MOVE: {
    if((destDir == null) || (destDir.equals("")))
     throw new ResourceException("Invalid Destination File
Directory");
    destFile = destFileName != null ? destDir +
destFileName : destDir + sourceFileName;

    closeFile(sourceFile);

    File destFileHandle = new File(destFile);
    File sourceFileHandle = new File(sourceFile);
    boolean status =
sourceFileHandle.renameTo(destFileHandle);
    if(! status)
     throw new ResourceException("File can not be moved");

    String renameTo = destFileName != null ? destFileName
: sourceFileName;

    addWork(sourceDir, sourceFileName, destDir, renameTo,
WorkItem.MOVE);
    break;}
                             ...
                             ...
   }
  }
  catch(IOException ie)
  {
   throw new ResourceException(ie.getMessage());
  }
  return true;
 }
```

### Listing 4

```
public WorkItem(String sourceDir,  String sourceFile,
String destinationDir, String destinationFile, int
workName, ManagedConnection mConn) {
  this.workName = workName;
  this.sourceDir = (sourceDir.endsWith(File.separator)) ?
sourceDir : sourceDir + File.separator;
 ...
 ...
    }

    public void executeCommit() throws ResourceException {
  switch(workName) {
   case CREATE: break;
   case UPDATE:
   case DELETE: //delete the backed up file
    String fullNameWithPath = new String(destinationDir +
destinationFile);
    File toBeDeleted = new File(fullNameWithPath);
    closeFile(fullNameWithPath);
    if (!toBeDeleted.delete()) throw new
ResourceException("Clean Failure: failed to delete ... " +
toBeDeleted.getPath());
    break;
   case MOVE: break;
  }
    }
```

### Listing 5

```
public void start(Xid xid, int flags) throws XAException {
  if (mConn.localTransactionCCI != null) {
   throw new XAException(XAException.XAER_OUTSIDE);
  }
  activate(xid);
  BitSet states = null;

  switch (flags) {
   case TMNOFLAGS:
    states = getTransactionState(associatedXID, true);
    break;

   case TMRESUME:
    states = getTransactionState(associatedXID, false);
    states = updateState(states, SUSPENDED, false);
    break;

   case TMJOIN:
    states = getTransactionState(associatedXID, false);
    break;
  }
  XIDStates.put(associatedXID, states);
    }
```

### Listing 6

```
  public int prepare(Xid xid) throws XAException {
  BitSet states = getTransactionState(xid, false);
  boolean suspended = currentState(states, SUSPENDED);
  boolean markedForRollback = currentState(states,
MARKED_FOR_ROLLBACK);
  boolean readOnly = currentState(states, READ_ONLY);

  if (suspended) {
   throw new XAException(XAException.XAER_PROTO);
  } else if (markedForRollback) {
   throw new XAException(XAException.XA_RBOTHER);
  } else if (readOnly) {
   return XA_RDONLY;
  }

  states = updateState(states, PREPARED, true);
  XIDStates.put(xid, states);
  return XA_OK;
 }
```

### Listing 7

```
public void commit(Xid xid, boolean onePhase) throws
XAException {
  BitSet states = getTransactionState(xid, false);
  boolean prepared = currentState(states, PREPARED);
  if (!onePhase) {
   if (!prepared) {
    throw new XAException(XAException.XAER_PROTO);
   }
  }

  ArrayList workList = (ArrayList)XIDWork.get(xid);
  executeActionsList(workList, true);
  removeTransationHistory(xid);
    }
```

### Listing 8

```
public void setWorkingFolder(String workingFolder)
    {
       if (workingFolder == null ||
workingFolder.equals("")) throw new
RuntimeException("Invalid value for Working folder in the
Deployment Descriptor.");
       File workingDir = new File(workingFolder);
       if (!workingDir.isDirectory()) throw new
RuntimeException("Working folder specified is not a valid
Directory.");
       if (!workingDir.canWrite()) throw new
RuntimeException("Working folder specified does not have
write permissions.");
       this.workingFolder = workingFolder;
    }
```

### Listing 9

```
public void setWorkingFolder(String workingFolder)
    {
       if (workingFolder == null ||
workingFolder.equals("")) throw new
RuntimeException("Invalid value for Working folder in the
Deployment Descriptor.");
       File workingDir = new File(workingFolder);
       if (!workingDir.isDirectory()) throw new
RuntimeException("Working folder specified is not a valid
Directory.");
       if (!workingDir.canWrite()) throw new
RuntimeException("Working folder specified does not have
write permissions.");
       this.workingFolder = workingFolder;
    }
```

Jason Bell
J2SE Editor

# Lift Your Vision Higher!

Having ridden the storm of the dot-com decline, it's nice to see the worldwide press having a semi-upbeat tone about the tech economy. Java, as a language, rode the crest of the wave; it could do no wrong and Java developers were the geeks among geeks. We now sit and watch the ups and downs of our industry; we watch the continual bickering and arguments that crop up. There are times I actually question why I do any of this coding/consulting thing at all.

Recently I found myself stuck in a rut, doing the commute, doing the job, and going home. I'd died on the inside and didn't know which way to turn. I was enjoying programming but I felt it had no real value. Now I don't really hear developers talk about this sort of thing at all and there seem to be shocked faces when someone leaves an organization, like Bill Joy leaving Sun, for example.

I was faced with some choices. I could stay and continue with my work, stay but change my role, leave and move onto something similar to what I do, or, the real risky alternative, leave and do something completely different. It's easy to dream a dream but it's another thing to step out and do it, so with the support that I needed around me I handed my notice in.

Now I don't advise everyone to do what I did, but there comes a time when you have to look at your desk and ask yourself, "Am I really fulfilling my potential here?" For me the reality was that I was not and it was time to go.

The apostle Paul spent a lot of time encouraging Christians and the Church to go that extra step in their conduct. As Java programmers we have to encourage each other and the groups that we belong to (user groups, blog communities, IRC, and mailing lists). Continually ask yourself how you can improve your skills, programming attitude, and business conduct. Java is a life lesson; how do you think you are getting on?

I got hold of a couple of books, *Change Activist* and *Soultrader* by Carmel McConnell, that discuss how to take control of your working and personal lives and do something with them. They were an eye opener and made me realize that I was doing the right thing by moving on.

The next step is to think big. Not just a little bit more than what you already have, but *big*, way beyond your comfort zone. I'm sure there are people at Sun who walk around dreaming up all sorts of stuff that never sees the light of day. It doesn't stop them; the chance of developing that one thing that might make it is enough for them to carry on.

I am not going to compare Java to Christianity but there are some parallels. We have to stay focused on what is close to our hearts. Nehemiah was constantly disrupted from building the wall in Judah. Even when the laborers were weak, they continued building. Though the dot-com boom ended, the developers kept building.

Christians mature in their work; so do developers. Constant learning, empowering, and encouraging is vital to the growth of the Java language, and the development of the application server and associated tools. We all have our part to play and watching public blog beatings only fills my heart with dismay as I watch the others who are trying to write more tools, and encourage and empower us to create more powerful applications.

I'm really excited about the next wave of applications, sites, and mobile devices and how Java fits in all that. I'm excited about getting out in 2004 and hopefully meeting some folks in the Java world.

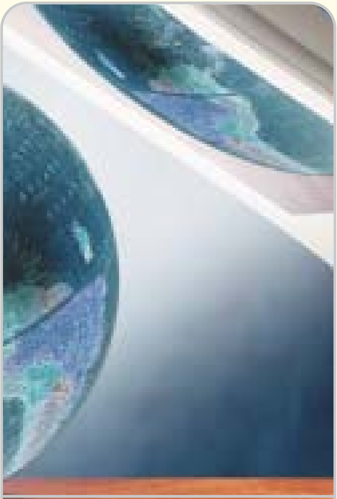
As we near 2004, here are a few questions to ponder:

1. Are you really happy with what you are doing?
2. If not, what are you going to do to change it?
3. If not, when are you going to change it?
4. How are you going to prevent it from happening again?

Regardless of everything you read on forums, Java is a top-class language. It runs fast and does the job wonderfully. Now let's get together as a body of developers who, with one voice, can show the world what this language can really do!

**Jason Bell** is the senior programmer for a B2B portal. He's also a keen supporter of people reading the API docs before asking questions. In his spare time he's involved with building RSS development tools.

*jasonbell@sys-con.com*



# Using **Java Generics**

## Create more flexible classes

Steve Close

Have you heard? Generics will be in the next release of the Java SDK (code named Tiger, aka JDK 1.5). You might be wondering "What is a generic?" or "Why should I care?" or even "Cool! How do I use them?" This article will introduce generic coding, explain how generics are used and what their advantages are, and discuss how they will impact your work. To help you understand, I'll define generics and code a few examples to illuminate how to use them.

Generics are not a feature that everyone has used. A concept similar to generics was included in C++, i.e., templates. Although the syntax of Java generics is modeled after C++ templates, the Java syntax is easier to understand. In addition, the implementation of templates and generics are not the same. Java remains type safe and doesn't expose source code when supporting generics. In other words, Java adds the power of generic coding without many of the problems found in other language implementations.

### What Are "Generics"?

Generics are known by another name that makes more sense to the average coder. They're also called parameterized types. Using parameterized types allows you to define a more flexible typed class or parameter so you get more type-checking support from the compiler. They solve one of the most annoying things in Java coding – the constant conversion of object references to a more useful reference type. This problem is especially prevalent when using collections of objects.

To start learning how to use generics, we'll look at some collections-based examples and see what the advantage is.

### Generic Examples

When working with any collection type, you're working with an object that can store objects, right? Actually you're working with an object that can store java.lang.Objects, but we don't need to be picky. If you look at Listing 1 you can see that every time you want to access an object in an ArrayList, you need to cast that object type. This listing is a remedial example of the need to cast all objects returned from an ArrayList. In this case nothing is done with the returned string, but it must be cast in order to access its string functionality.

In comparison, when using generics your code would look a little different (see Listing 2). Instead of creating an ArrayList reference, create an ArrayList reference to an ArrayList that can hold only strings (lines 9 and 10, Listing 2). The compiler will only allow strings to be added to this ArrayList, so any attempt to add objects of any type that is not a string will generate a compiler error. In addition, when you retrieve any value from the ArrayList, it's returned as a string so no cast is needed (line 14, Listing 2).

The collections you're familiar with in JDK 1.4 have been rewritten in 1.5 to handle generics. The good news is you won't have to fix or convert your existing code. Existing code will work with no changes in most cases. In fact you can still use the old form of all your favorite collections, but I'm not sure why you would, unless you're stuck on pre-1.5 Java.

### Why Do We Need Generics?

We don't really need them, but they add to the language and provide distinct benefits. Using generics reduces the need to cast references back to the actual types of the object, one of the ugliest little code snippets in all of Java. As a bonus, the frequency of ClassCastExceptions should be reduced because the compiler will catch type errors, and the need to surround casts with "if ( o instanceof )" is also gone. Errors caught by the compiler are a good thing; runtime exceptions are not as good. The addition of parameterized types allows you to create code that is geared to collections of objects of a single specific type.

The following code demonstrates how a method can expect a collection of Integers, not just a collection. The author of this method doesn't need to check type or cast, and the code is cleaner because of this. This static method filters all values larger than a given cap out of any given collection of Integers.

```
public static void capList
(Collection list, int cap)
{
Iterator itor =
list.iterator();
while (itor.hasNext()){
if(itor.next().intValue() > cap){
itor.remove();
}
}
}
```

Notice there's no need for the author of this method to cast Object to Integer; the compiler will take care of it. The author of the code defines what type of collection can be used. If a user of this method attempts to pass the wrong type of object, it will be refused by the compiler, giving a "cannot be applied" compiler error. The code below, written to invoke the "capList()" method, contains the compiler error.

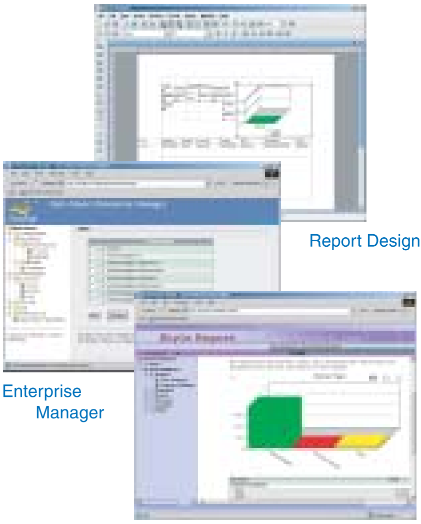
```
LinkedList list3 =
new LinkedList();
list3.add("8");
//The next is a compiler error
capList(list3, 30);
```

Rewriting the same method without a generic collection would require you to cast each element retrieved from the collection. This in itself is not terrible (heck, we've been doing this in Java for quite a while), but the generic example is cleaner. The version that includes generics is cleaner and the

# Java & Web Reporting:
# with Java flexibility
# without Java complexity

## Style Report:
## Information Vantage Point

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

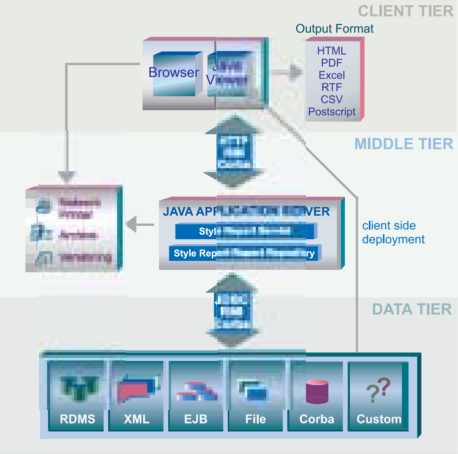


Report Designer

Enterprise Manager

Web Report Viewer/Composer

STYLE REPORT ARCHITECTURE DIAGRAM

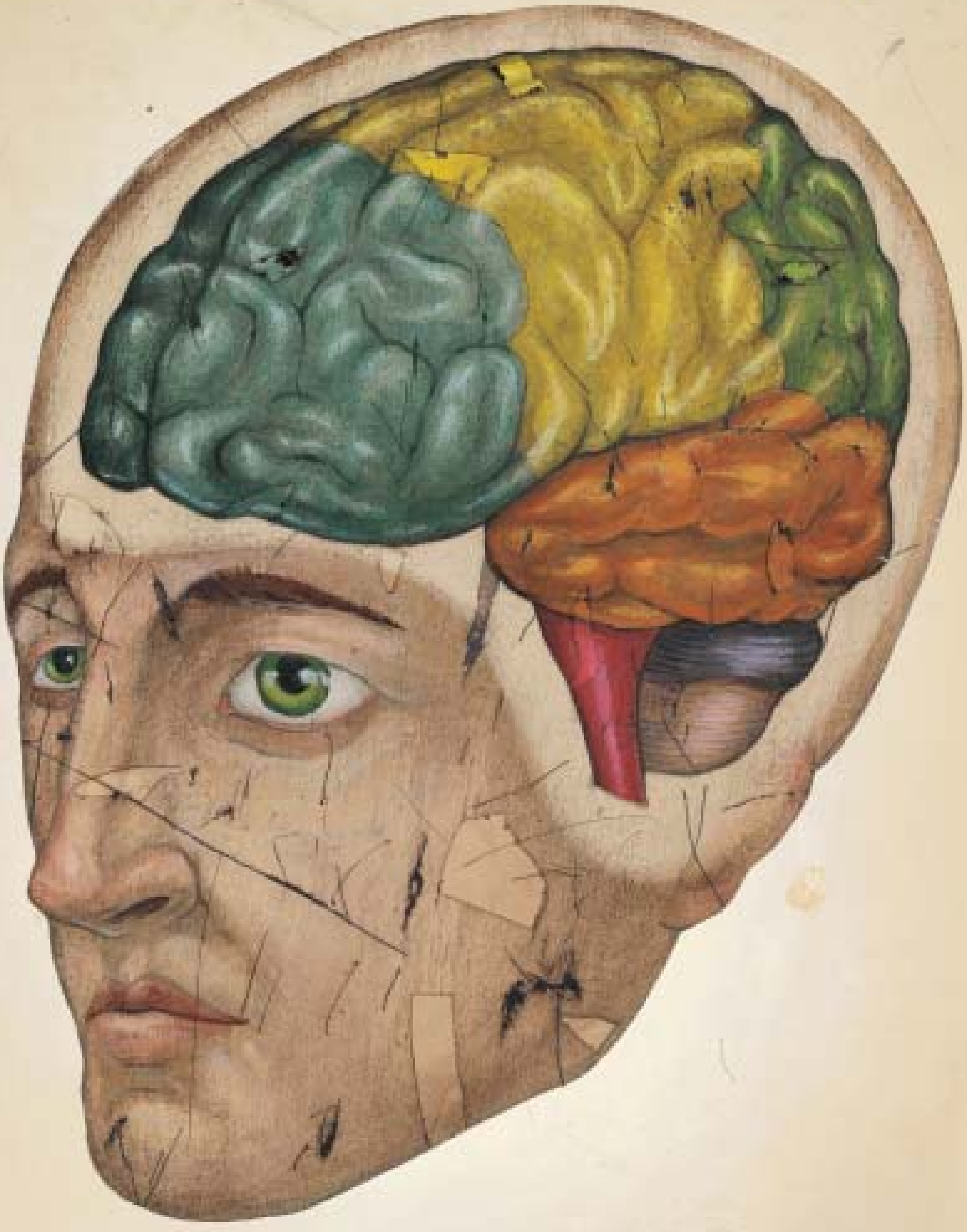


| Style Report Features | |
|---|---|
| Flexible Visual Designer with JavaScript or VB Script | ✓ |
| PDF, Excel, Word, CSV and other Export Format | ✓ |
| Database/XML and other Datasources | ✓ |
| Zero Client Browser Viewing | ✓ |
| Built-in Drilldown/Parameters | ✓ |
| Zero Client Ad-hoc Reports |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

InetSoft
open standards innovation

www.inetsoft.com
JDJ special http://www.inetsoft.com/jdj

Phone: 888-216-2353 (USA)
Phone: 732-424-0400 (Intl)



J2ME | J2SE | J2EE | HOME

methods declaration indicates to the user of the method what is expected as a parameter. The code itself, rather than documentation, leaves little question regarding what the collection can contain.

For comparison, the same method is included below without the addition of generics. Notice the cast in the if-statement and the use of instanceof to check the type before using it as an Integer.

```
public static void capList
 (Collection<Integer> list, int cap){
Iterator<Integer> itor = list.iterator();
Object o = null;
while (itor.hasNext()){
 o = itor.next();
 if ( o instanceof Integer &&
((Integer)o).intValue() > cap){
itor.remove();
}
}
}
```

Most of the generics changes that your average Java programmer will see are in the collections API and in extending those collections. The collections become easier to use and the compiler supports typing, as we saw above. Gone are the days when you created your own collection type to hold only one type of object; generics makes that code unnecessary.

The way generics change collections is not the end of the story. Code designers will be using generics to make code more flexible in more than just collections. If the generics change had only changed collections, Java would have been better off adding a dynamic array type.

You'll also be able to work with generics in your own code. The syntax is new to non-C++ converts, but it's simple to work with. As an example we'll create a class that has a parameterized (generic) property, a property that can be of any type. The class in Listing 3 is a standard JavaBean with a getter and setter, but the author of this class doesn't need to know what type is being stored. The property type is generic, so the author has inserted a < > after the class name. In this < > notation is a variable type name; it needs to be a valid Java identifier or a comma-separated list of valid identifiers. After assigning a Java identifier between the < >'s, you can use that type name in your class the same as any other type name. The actual type that it will be is defined by the user of the code in Listing 3.

**Steve Close** has been working as a Java consultant and trainer since 1997. He has authored seven popular workshops for Intertech on topics ranging from Complete Java Programming to Expert J2EE Patterns. He served as president of the Twin Cities Java Users Group from 1997 to 2000.

*sclose@intertech-inc.com*

When the client creates the bean, he or she will instruct it to hold some object type. From that point on the compiler will make checks for type safety; for example, when one is created with the following statement:

```
GenericBean<String> bean = new GenericBean;
```

After the reference to the GenericBean is defined as holding strings, the compiler will allow only strings to be sent into the setter, and the getX() method will always return a string. This flexibility and type safety adds up to easier code to write and to read.

## Generics Issues

Generics represent a fundamental addition to Java syntax. Changes at such a low level do not come for free. Unlike adding new APIs or adding features to JDK back in the day, this change will need to be compatible with millions (billions?) of lines of code. There are a lot of existing Java applications and the switch to Tiger might cause a few errors in your existing Java code.

In a recent JavaOne technical talk, Gilad Bracha (lead of JSR 14) predicted that one out of every million lines of code will have compiler errors. This prediction was based on the two errors found when compiling the 2+ million lines of code in the JDK. For my business 1 in a 1,000,000 seems pretty trivial, but I don't have 5,000,000 lines of mission-critical legacy Java to support.

The JCP expert group for generics worked hard to minimize the impact the addition of generics can cause. Because compatibility was a design constraint, a few annoying features have been accepted into generics, including new type casting possibilities. Specifically, you can still get class cast exceptions when working with generics. This is possible because a reference to a type can be stored in a reference that doesn't appear to be parameterized. Whoa, that might sound a little better in code. The following example shows an ArrayList stored in an ArrayList reference. The new reference to the array list called list5 allows you to add non-strings to the ArrayList.

```
//This snippet will compile fine
ArrayList<String> list4 =
new ArrayList<String>();
list4.add("Steve");
ArrayList list5 = list4;
list5.add(new Object());

list4 = list5;
```

This issue can't be fixed because it's needed to make the new collections backward compatible with any code previously written, but it can cause trouble. Another way to put it is "It's the better of two imperfect options." The compiler allows this change, even though it will most likely cause a ClassCastException. This situation will probably occur rarely, typically when you're changing existing code that's used by other parties, but the problem exists. The compiler will inform you that you might have an issue by sending a warning. If you compile with the -warnunchecked flag, it will provide details. The above snippet produced the following warnings.

**Listing 1**

```
package example1;

import java.util.ArrayList;
public class OldCollection{
public
static void main(String[] a){
ArrayList strLst =
 new ArrayList();
 strLst.add("hello");
 strLst.add("goodbye");
 //...
 //Object reference
 String strFromList =
 (String)strLst.get(1);
 }
 }
```

**Listing 2**

```
package example1;

import java.util.ArrayList;

public class UseGenerics{

public
static void main(String[] a){
ArrayList<String> strLst =
 new ArrayList<String>();
 strLst.add("hello");
 strLst.add("goodbye");
 //...
 String strFromList =
 strLst.get(1);
 }
 }
```

**Listing 3**

```
class GenericBean<VarType>{

private VarType x;

public GenericBean(){}
public GenericBean(VarType x){
this.x = x;
}

 public VarType getX(){
 return x;
 }
 public void setX(VarType x){
 this.x = x;
 }
 }
```

```
warning: unchecked call to add(E) as a mem-
ber of the raw type java.util.ArrayList
list5.add(new Object());
warning: unchecked assignment:
java.util.ArrayList to java.util.ArrayList
list4 = list5;
```

Another annoying feature is the limitation on types. Generic types don't support primitives. You still can't make a collection hold int, char, or any of the primitives. The following code would fail to compile.

```
//This doesn't work ArrayList<int> intArray
= new ArrayList();
```

The annoyance here is an old one. Java separated out primitives from objects to improve performance and that's a good thing, but it also forces developers to learn about "wrapper" types. If you want to store primitive ints in a collection, put them into java.lang.Integer objects.

I mention this annoyance because I hope that JSR 201 will be included in JDK 1.5, which would allow you to ignore the wrapper classes with the automatic "boxing" of types. If you want to give it a try, the code for boxing in Java appears below. Notice that the ArrayList type allows you to add ints without making Integers. The Integer is created behind the scenes. Boxing is a feature of C# and VB.NET that makes me just a little jealous, but they don't have generics yet.

```
LinkedList list2 =
new LinkedList();
list2.add(8);
list2.add(43);
list2.add(4);
```

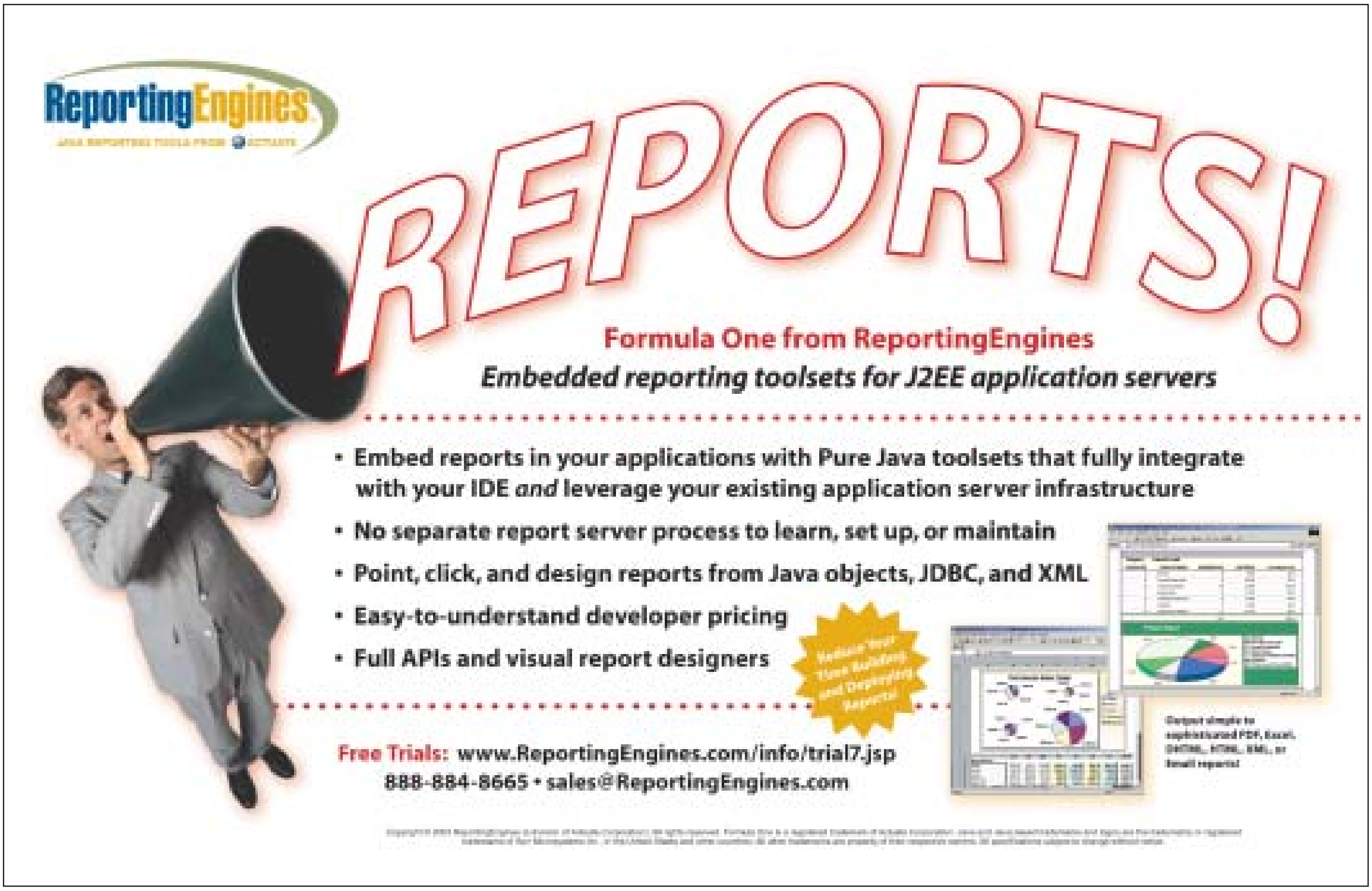
## Trying It Out

If you want to try out generics, you don't have to wait for Tiger. A reference implementation is available now that works with JDK 1.4.1. This implementation isn't difficult to use or set up. To get it going you'll need a little knowledge and two downloads. You need JDK 1.4.1, not JDK 1.4. You can find it at http://java.sun.com; download and follow the installation directions. You'll also need an implementation of the generics compiler and the new definitions for the collections. These are contained in a zip file called adding_ generics-2_0-ea. zip, which can be downloaded from Sun at http://developer.java.sun.com/ developer/early Access/addinggenerics/ index.html.

Finally, you'll need to register as a member of the Sun Developer Network. Joining is painless. After downloading the zip, unzip it to a folder. I downloaded it onto my Windows box and unzipped it to c:\java. The zip contains a folder named "adding_ generics-2_0-ea" that I'll call JSR14HOME when describing the commands needed to compile and run your generics examples. The generics download contains JSR 14 features (Generics) as well as JSR 201 features (enumeration, boxing, static imports, and the "for-each" enhancement). Sample code is included as well as the source code for many of the changes to existing collections. There is a .bat file to help compile in the scripts directory, but I wrote my own because it was untested and required adding environmental variables I would never use again.

When compiling you need to include a few extra arguments for javac. The following is the javac statement I used to compile. JSR14HOME refers to the location of your adding_generics-2_0-ea folder and the JDK141Home is the Java_Home directory for an install of the JDK version 1.4.1.

–continued on page 60



# An Interview **with Bruce Eckel**

Interviewed by
Jason Bell

Recently, Jason Bell had the opportunity to talk with Bruce Eckel, noted author of *Thinking in Java* and *Thinking in C++*.

**JDJ:** *Thanks for taking the time to talk with us. I know you've recently had some seminars in Prague. Do you think European programmers differ from American programmers?*

**Bruce Eckel:** That's very difficult to say, since I believe we tend to get a special group of programmers at the seminars there. Even though we offer the seminar at lower prices, it's still expensive and as a result those who do come seem to be the exceptional ones. They tend to be outstanding, but I assume there are other factors involved.

**JDJ:** *Do you have any comments on the new additions to the JDK 1.5?*

**Eckel:** I assume you mean the features that mimic those in C#. I'm very glad to see those, because it means that the Sun Java folks are no longer saying "you don't really want those things." Instead, they're stepping up to the competition right away, which is very promising. I was very skeptical about C# at first, but when I examined it more closely I realized it was a well-designed language.

**JDJ:** *I know you use a number of languages, but which one would you suggest for beginners who want to learn object orientation?*

**Eckel:** Python. It's the perfect beginner language, and it's built on objects from the ground up, so it's easy to learn to use objects with the language. Unlike Java or C++, there's very little ceremony required, so you don't get distracted by weird artifacts and can focus on the essence of objects and programming.

Then if you want to move to Java or C++, the concepts translate nicely and you can easily see what the object's essence is in that language and which are the language artifacts. And you can continue using Jython in various ways for your Java development. Also, there's Boost.Python to make it easy to talk back and forth between Python and C++

**JDJ:** *I've spoken to a lot of programmers over the years and I've heard nothing but praise for* Thinking in Java. *Are you surprised by its success?*

**Eckel:** I guess I am. I've been writing books so long that I don't expect them to necessarily do that well (first-time authors often expect to retire on book royalties, but after you've done it awhile you discover that a "successful" book in the publisher's eyes is often just one that pays back its advance). But since *Thinking in C++* (when I started using automated code testing tools to validate the code in the book, in 1995) I also expected that everybody would naturally start using automated testing tools for their book code. At this point, I've still met only a couple of authors who do it. Also, I follow John Irving's maxim: "Writing is rewriting." But that's pretty much all I do – make sure my code works and rewrite a lot. As a result of rewriting, I find a lot of things in the books that I consider embarrassing, so I always think they could be a lot better.

**JDJ:** *Do you offer the code updates on your Web site?*

**Eckel:** Yes. The code and the HTML version of the book is available for free download from www.MindView.net. There aren't a lot of changes between versions of the book, but if I do make any changes I post them on the site.

> "If you want to move to Java or C++, the concepts translate nicely and you **can easily see what the object's essence is in that language"**

**JDJ:** *I notice you have a Web log. What made you decide to run one? Do you get time to read any other programming blogs?*

**Eckel:** If you go to the first Web log at http://mindview.net/WebLog/log-0001, titled "The origin of this Web log," that will answer your first question. Bill Venners got me started, and even though I didn't end up writing on his site, I still follow his logs a bit. He's amassed quite a collection of quality Web loggers, and he focuses on programming.

For years I've read Elliotte Rusty Harold's "Cafe Au Lait" Web site (www.cafeaulait.org), which is kind of like a Web log for Elliotte but better – it's daily and he hunts for interesting articles, so it's as if he's filtering the Web for me, especially Java stuff. I rely pretty heavily on the Elliotte-bot to keep track of the interesting stuff. I occasionally read some columnists, like Cringely (www.pbs.org/cringely) and Joel Spolsky (www.JoelOnSoftware.com), and www.pcmag.com columnists. But these are when I'm killing time. Usually when I'm at the computer I try to get something done.

**JDJ:** *In one of your blog entries it said "If it's not tested, it's broken." Do you have any words that would encourage nontesting programmers to test their code?*

**Eckel:** That's actually a quote from *Thinking in Java, 3rd edition*. There's a section in Chapter 15, "Discovering Problems," that attempts to convince people of the value of testing. Also, the talk I give to user groups lately is about testing (unit testing and design by contract, also in *TIJ3*). But in general it seems to be a personal epiphany that each programmer must have, when they suddenly see that automated unit testing saves them a lot more time than it costs. This happens when you change something in your code and your unit tests suddenly find a lot of bugs that you know wouldn't have shown up for a long time otherwise. That's convincing – after that, you can't imagine doing without.

**Bruce Eckel** (www.BruceEckel.com) is the author of *Thinking in Java* (Prentice-Hall), the *Hands-On Java Seminar* CD ROM (available on his Web site), and *Thinking in C++* (PH), among others. He's given hundreds of presentations throughout the world, published over 150 articles in numerous magazines, was a founding member of the ANSI/ISO C++ committee, and speaks regularly at conferences. He provides public and private seminars and design consulting in C++ and Java.

*JDJ: Are there any books you can recommend (apart from your own) that have been a help to you?*

**Eckel:** *Design Patterns*, of course; despite its problems it's still a major learning experience. Gerald Weinberg's *Secrets of Consulting* is a must, as is *Peopleware* by DeMarco and Lister (2nd edition, and they just came out with a new book, *Waltzing with Bears,* that I'm sure is good). *Effective C++* and *Effective Java* have both been quite helpful. I finally understood network programming from Elliotte Rusty Harold's *Java Network Programming*. *Learning Python*, *The Python Standard Library*, and *The Python Cookbook* are all great. *Core Java* has always been helpful as a resource. Some of the *Extreme Programming* books have been good. I generally like Robert Glass's books for their irreverence about software development. A lot of Martin Fowler's writing is quite helpful. That's all that I can get from skimming over my bookcase.

*JDJ: With everything you do, do you have time for any non-work activities?*

**Eckel:** Definitely. That's one of the main points of working for myself – the freedom to take time off and do other things. One of the activities I enjoy a lot when I'm here in Crested Butte, Colorado, is mountain biking. I'm competent in that I can cover the territory (sometimes that means walking the bike) but it's excellent exercise and a great way to be outside. Also hiking, since the mountains are beautiful here. I usually take people on hikes when they come up here for seminars.

Also, on a lark I just tried out for a local play here, however, I don't know if I got the part, but it turns out it was a lot of fun just to read for it.

*JDJ: What do you have planned for the remainder of 2003?*

**Eckel:** We aren't doing that many public seminars since people haven't been that keen to travel, and the economy has definitely impacted training budgets (although I have seen some small indications that this trend has at least bottomed out, if not slightly turned around). I suspect we'll have at least several in Crested Butte before the end of the year, but the rest of the time I will be working on books (finishing *C++, Volume 2*, starting a new book that I haven't announced yet) and the "Hands-On Java CD ROM," finishing the solution guide for the third edition. I also hope to set up a simple studio to create video training media, which is something I've been pursuing on and off since I worked on *The World of C++* and *Beyond the World of C++* many years ago for Borland (see http://mindview.net/WebLog/log-0029 for details). Especially with people not traveling and with budgets the way they are, it seems as if CDs and videos are a great way for people to learn now.

Also, I've been doing audio interviews with various luminaries in the software world that I plan to make into an MP3 CD for people to listen to while they drive (with the new inexpensive MP3 players) or on their computers. These interviews have been quite interesting so far; I'm going for the NPR "Fresh Air" kind of feel. They've also been fun to do, which I think is a good sign.

*JDJ: Dare I ask which luminaries you have interviewed? Just as a teaser…*

**Eckel:** I'm going for a spectrum of people involved in different aspects of creating software. Not everyone is well known. Some are book authors like Dave Thomas, Andy Hunt, Joshua Bloch, and Ron Jeffries. Chuck Allison wrote a C/C++ book and is coauthoring *Thinking in C++, Volume 2* with me, but his focus is teaching so we talked about the education necessary for good software engineers. Guido Van Rossum created the Python language and Jim Fulton created the Zope application server. Gene Wang used to be VP of languages at Borland, and is now president of a company that makes cellphone software, so we mostly talked about management. Daniel Will-Harris is a designer (he does my book covers) and has consulted on the user aspects of software; we discussed, from the user's perspective, the kinds of things that software designers need to do in order to create better software.

There are others that I hope to add, and I also hope to continue doing interviews and creating CDs like this because I'm having fun doing it. And the questions I asked were things that I found interesting, so I think the interviews themselves will be quite enjoyable to listen to.

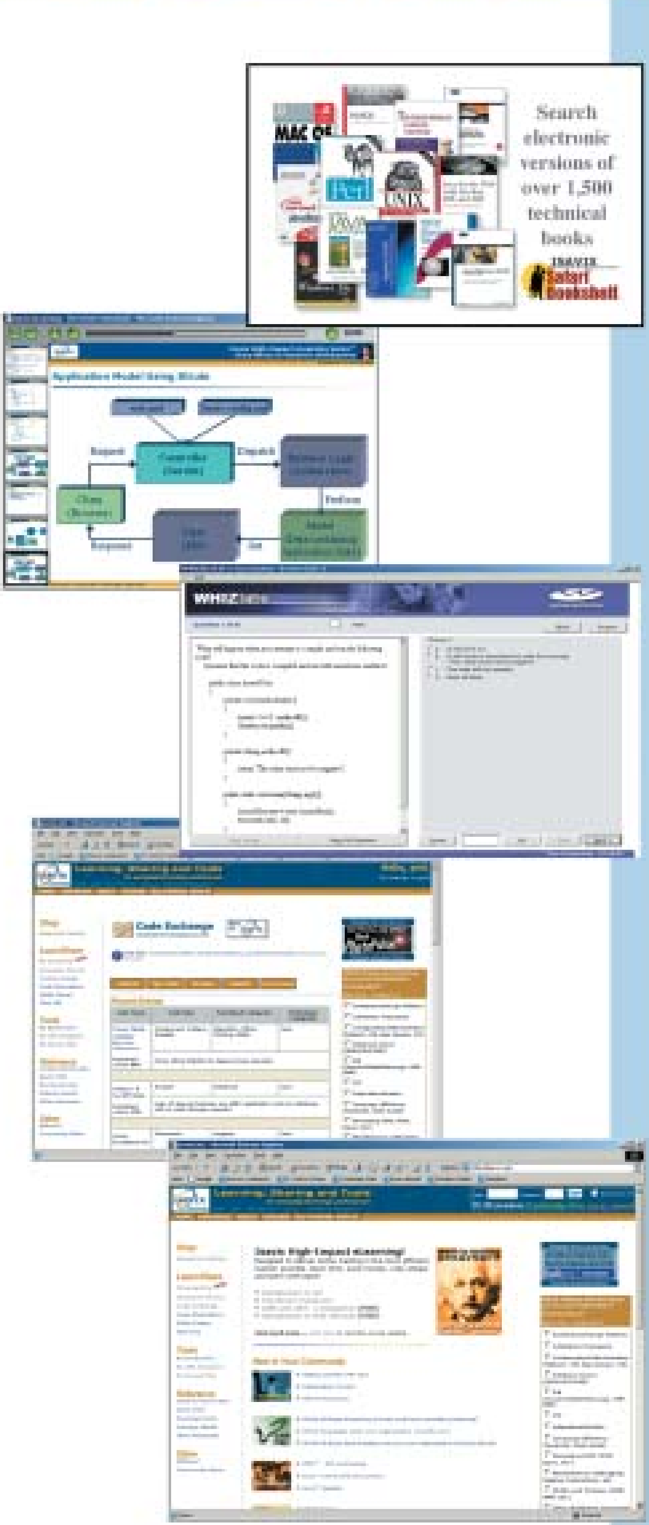
*JDJ: You mentioned MP3s; do you listen to any music while you're coding?*

**Eckel:** No, I'm not the kind of person who can usually listen to music and do complicated mental stuff at the same time. I had a college roommate who could only study with headphones and music, and I could never figure it out. On the contrary, I prefer things as silent as possible, and my father (a master craftsman in wood) just finished building me a desk for use here in Crested Butte that has a sound-insulated cabinet for the CPU box. Since my new computer is very quiet, the combination of the two should be completely silent.

I am interested in some of the new MP3 technologies, however. In particular, stand-alone systems that hook into your stereo. Maybe if I master a system like that, I'll experiment with listening to music while writing and see what happens.

*JDJ: Bruce, once again, thank you for taking the time to talk to us.*

**Eckel:** Well, I think it has helped kick start me back into my own writing. This was the easiest project (talking about myself) that I had on my desk.



Search electronic versions of over 1,500 technical books
ISAVIX Safari Bookshelf

J2ME | J2SE | J2EE | Home

# MIDI & audio sequencing with java

by Mike Gorman

The Java Sound API, first introduced in J2SE 1.3, includes the package javax.sound.midi, which contains everything you need to be able to send and receive messages to and from any MIDI device visible to your operating system.

The Java Sound Programmer Guide and the Java Sound Demo, both available for download from Sun, are excellent references that illustrate all the “nuts and bolts” of sending and receiving messages. This article provides a brief overview of working with the MIDI and sampled audio primitives of the Java Sound API, and then explores using those primitives to construct a basic multi-track MIDI/audio sequencer in Java.

## Programming with MIDI

Basically, every message you send to or receive from a MIDI device is one or more bytes. The first byte is referred to as the “message” or “status” byte. This is essentially the “command” (e.g., note on, note off, change patch, set volume, etc.). The specific command you are sending or receiving will determine what the next byte or bytes are, if any. Having an online MIDI specification reference available will be indispensable.

The first step is to obtain a reference to the specific MidiDevice you want to talk to. The javax.sound.midi. MidiSystem is your gateway to everything that Java Sound was able to detect was installed in your operating system. You might display a list of available devices to users and let them choose which device they want to use (see Listing 1).

Once you have a reference to your MidiDevice, you’re ready to begin sending and receiving messages. The first step is to understand that there are Receivers and Transmitters. As you might suspect, Receivers receive MIDI messages and Transmitters transmit or are the source for MIDI messages.

Step one is finding out which MidiDevices the MidiSystem is reporting. The easiest way is to iterate through each one, try to play “middle C,” and see what happens. You may find that some of the MidiDevices are configured as receive only, others are transmit only, and others might receive and transmit. In my setup, my MIDI interface reports two different MidiDevices for the same keyboard – one that is transmit only, the other that’s receive only. So when I want to send a MIDI message, I send it to the MidiDevice that is receive only, and when I want to record MIDI messages that I trigger by playing the keyboard, I do it by listening to the MidiDevice that’s configured to transmit only (see Listing 2).

Step two is to figure out which of the MidiDevices are transmitters. In this case, you’ll need to create an implementation of the javax.sound.midi.Receiver interface (see Listing 3).

Next, figure out which of the available MidiDevices are acting as your keyboard’s transmitter by trying them out one at a time. Obtain each MidiDevice’s “transmitter,” assign your receiver to it, play a few notes on your keyboard, and look for output in the console that indicates you’ve found the right “transmitter” device (see Listing 4).

Once you’ve figured out which MidiDevice is your receiver and which is your transmitter, and you know the basics of sending and receiving MidiMessages, you now have everything you need to create your own full-featured 16-track MIDI sequencer! Well, not quite…

## Creating Your Own MIDI Sequencer

As I soon found out, there’s a lot more to creating a sequencer than just knowing how to send and receive MIDI messages.

You may have noticed by now that the javax.sound.midi package already includes a class Sequencer. Unfortunately, this “built- in” sequencer is limited in its capabilities and is not extendable since it’s an interface. But the biggest reason I was unable to use it is that it seems to be “hard wired” to use the internal Java synthesizer (Sun Bug ID 4783745). It also appears to have very bad timing problems (Sun Bug ID 4773012).

With the API it’s easy to create your own sequencer. First we have to be able to record a single track and play it back in the exact same timing it was originally played in. Moreover, the performing artist (that’s you) will want to have a four-bar metronome count off prior to recording start, then, to keep perfect time, you’ll need to continue the metronome until the user clicks stop. Of course, the metronome sounds should not be part of the performance when it’s played back.

You can have the computer emit a “system beep” for your metronome, but I prefer to listen to the hi-hat of a drumkit on the keyboard. A lot of sequencers use MIDI channel 10 (i.e., track 10) as a drumkit, but you can choose any one you like. A tempo of 120 beats per minute (bpm) means 2 beats/second or 1 tick of your metronome every 500ms.

As you may have guessed, you’ll need one thread playing the ticks of your metronome (on Channel 10) while you record any MidiMessages you receive (via your Receiver implementation) on Channel 1. (Note that I am referring to the channels/tracks from the musician’s perspective. The channel references in the API are zero-based.) Listing 5 shows what your metronome thread might look like.

Playing the metronome is simple enough, but you need to figure out a way to play it through just one time, then begin recording MidiMessages as they arrive at your receiver (discussed later). How you do that will be left for you to decide.

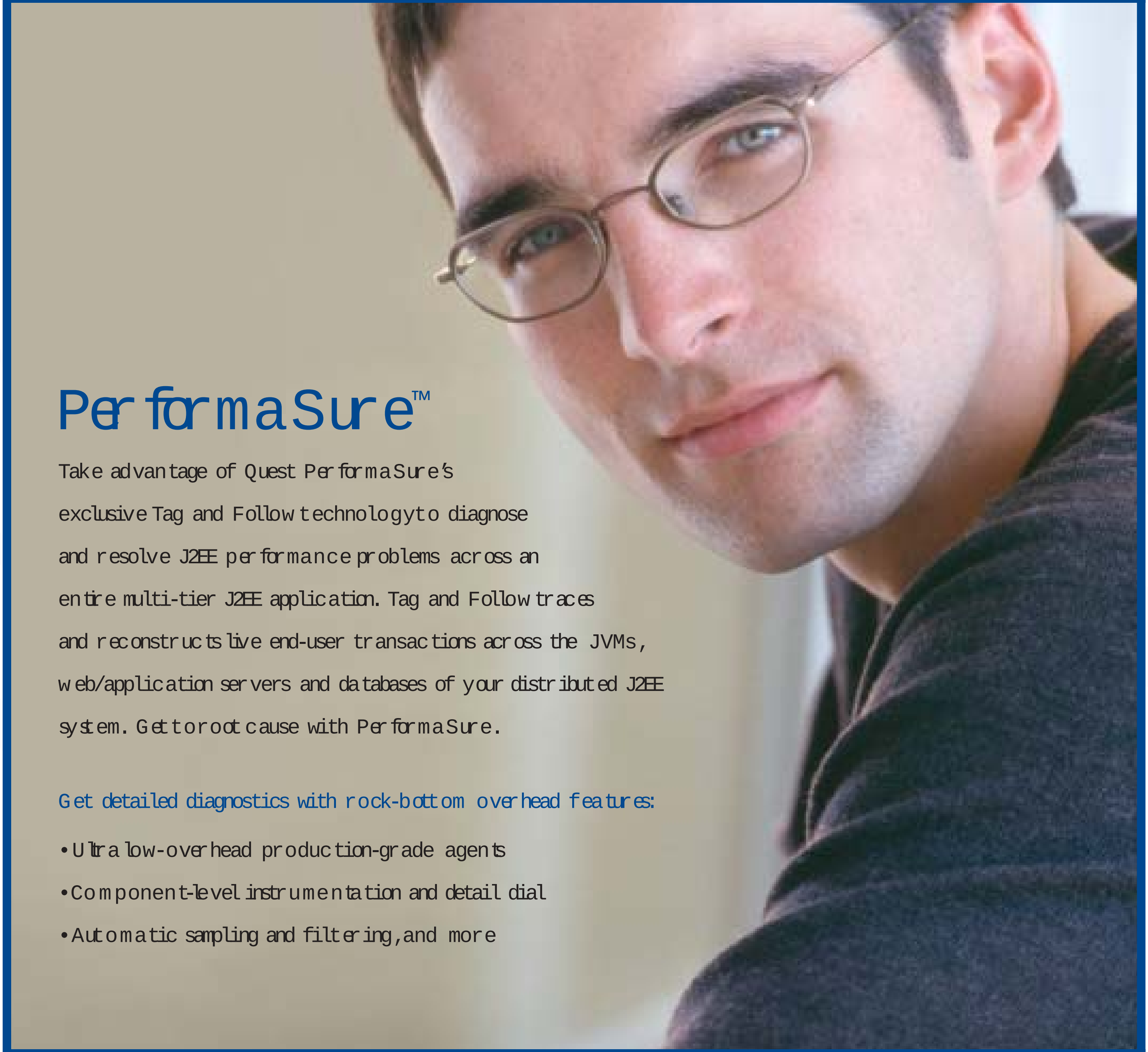
## Recording MIDI

How exactly do you record MidiMessages? There are basically two strategies: you can try to take note of what time each message arrives, or you can use the included timestamp of each message. In either strategy, your implementation of the Receiver interface will create an ArrayList and add each MidiMessage it receives to the ArrayList. Of course, you’ll need to make sure you record only MidiMessages for the duration immediately following the four-bar Metronome count off until the user clicks stop.

Your first strategy might be to use System.currentTimeMillis() to take note of the current system time (in ms) at which each MidiMessage arrives. You’ll need to know this when you play back these messages. The general idea is to play back the messages using a thread, that’s sleeping between messages, according to the relative time they originally arrived. In my experience, the system clock was not reliable enough to deliver rock-solid timings during playback. You’ll know what I mean if you try this strategy when you listen to the playback of messages based on the system clock.

The other strategy is to use the embedded timestamp that accompanies each MidiMessage. This timestamp is expressed in microseconds based on the time you first opened the MidiDevice. Unfortunately, by the time the four-bar metronome count off ends, it’s difficult to say when the first message should be played back. That is you can’t assume that the first message that arrives should be played back at time zero. Perhaps the musician’s first note is played halfway through the first measure. Since the MidiDevice was opened long before your metronome began playing, it’s difficult to determine from the timestamp alone how much time your playback thread should wait until it sends the very first message. Of course, all messages after that are easy, since you can just calculate the time to wait in between each message based on the relative differences of the message’s timestamps.

The best solution I came up with was to just take note (by way of System.currentTimeMillis()) of when recording actually begins (that is, after the four-bar metronome count off), and then take note of when the first MidiMessage arrives. Then, during playback, the playback thread merely needs to wait the



J2ME
J2SE
J2EE
Home

calculated delay time before playing back the first message. Thereafter, it can simply use the relative differences between the MidiMessage timestamps for all subsequent messages.

It may surprise you to learn that what you think of as a chord (or several chords across multiple tracks) struck simultaneously is actually played back one note at a time, sent serially as a stream of MidiMessages, one at a time. You have to remember that the playback loop playing back the messages is so fast that the human ear will not be able to discern the difference between the original "three notes struck simultaneously" and "three notes played 1 ms apart."

You should now be able able to record and play back a single MIDI track at 120 bpm. If, when it plays back, it sounds just like you played it, you're halfway there. The next step is to be able record additional MIDI tracks while playing back previously recorded tracks.

### *Recording Multiple Tracks*

You may have already begun to notice that, although you are receiving and recording the MIDI messages, it's hard to control what sound/voice/patch the keyboard is playing. This is why each of the 16 MIDI channels on the keyboard can have a different patch associated with it. Most keyboards allow you to change what MIDI channel they are transmitting on. Whatever MIDI channel you have selected on the keyboard should also change the patch selected as well.

The problem is that you don't want to constantly have to make sure your keyboard's selected channel matches the track you play to record in your sequencer. If they're not in sync, you'll think you're recording track/channel 2, but the keyboard still has channel 1 selected. Although you may have the "channel 2 ArrayList" full of the MidiMessages you received, those messages have one of their bytes indicating that they are channel 1 messages, and so playback of those "channel 2 messages" results in playback on channel 1, playing channel 1's patch instead of channel 2.

The solution seems tricky and not very efficient, but it seems to work just fine. The trick is to first turn off the keyboard's "keyboard" from triggering sounds internally; it will continue to transmit MIDI messages as usual:

```
ShortMessage msg = new ShortMessage();
msg.setMessage(
  ShortMessage.CONTROL_CHANGE, 122, 0);
_receiver.send(msg, -1);
```

Next, "route" all incoming MIDI messages to the keyboard, playing them back on the track the user thinks he is recording. For example, you may receive all your MIDI messages with the "channel 1 byte" set. If the user thinks she is recording track 2, then for each MIDI message received, in addition to recording it (by storing it in track 2's message ArrayList), change the "channel byte" to 2 and retransmit them back to the keyboard (see Listing 6).

### *Playing Back Multiple Tracks*

Assuming you have several different tracks of MIDI data recorded, it's time to play them back. Your first approach might be to use a separate thread for each track (channel). While this is an intuitive programming model, you'll quickly find that although each track (thread) plays back in perfect time relative to itself, it's difficult to keep it perfectly in sync with the other tracks. If your tracks are short and you plan to loop them, you could use thread synchronization to make sure all tracks "sync up" with each other at the end of each iteration. However, you will soon find your clean sequencer code is getting cluttered up with complex thread synchronization all over the place, and it becomes harder and harder to manage and still achieve "rock solid" timing.

What I found to be easier to manage and virtually guaranteed to stay "in time" was to collect all MidiMessages, regardless of track (channel), put them into a single ArrayList, sort them all based on their timestamp, and then play them all back using a single playback thread.

## Adding Digital Audio

By now you should have a good instrumental recorded using multiple MIDI tracks, but you'll add more interest to your song by laying down a vocal track or two. Luckily, the Java Sound API includes the javax.sound.sampled package dedicated to recording and playing back digital audio.

### *Recording Audio*

Ultimately, any recorded digital audio comes down to samples. A sample is a measurement at a point in time of what you might picture as the audio "waveform." The standard CD sampling rate is to take 44,100 measurements, or samples, each second. Each sample may be 8 bits, 16 bits, or more. There are a variety of sample formats in use today, and the Java Sound API supports about everything you'll encounter. Some useful constants for recording CD quality sound are:

```
AudioFormat.Encoding encoding =
  AudioFormat.Encoding.PCM_SIGNED;
int rate = 44100;
int sampleSize = 16;
int channels = 1;
boolean bigEndian = true;
```

An AudioFormat object will be needed later:

```
AudioFormat format = new AudioFormat(
encoding, rate, sampleSize, channels,
  (sampleSize / 8) * channels, rate,
  bigEndian);
```

Before you can begin recording, however, you'll need to obtain a TargetDataLine. The Java Sound API models its sampling API in terms of "lines." A line may be a microphone input, a previously recorded sample, the computer's "line out" or speaker, or any type of "input" or "output." To facilitate the playback of multiple samples at the same time, the interface Mixer is provided, which is itself a type of line. Lines may have controls that parallel what you'd find in a real mixer – gain, pan, volume, reverb, equalization, etc.

Like the MidiDevices returned from the MidiSystem, the class AudioSystem serves as your gateway into finding out and obtaining whatever Lines and Controls are installed and available to you. In general, the first step to recording an audio track is to obtain a TargetDataLine suitable for recording audio in the format requested, in this case an AudioFormat that is a single 16-bit channel recording 44,100 samples/second (see Listing 7).

As you may have suspected, you'll need a separate thread to capture the incoming sample data. Using the TargetDataLine and OutputStream created previously, you'll want to create a loop that reads a chunk of bytes at a time from the TargetDataLine, writing them out to the OutputStream until there's nothing left to read or until the user clicks stop (see Listing 8).

At this point, your ByteArrayOutputStream contains a ton of bytes. The average 3:30 minute song will require 9.3MB worth of samples for just a single mono track! FileOutput-

WebAppCabaret™

J2EE Web Hosting

www.webappcabaret.com

Quality Web Hosting at a reasonable price...

# <JAVA WEB HOSTING AND OUTSOURCING>

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

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

**At WebAppCabaret you have the flexibility to choose the LATEST hosting technology best suited for your WEB application or your programming skill. If you are a programmer or consultant,WebAppCabaret has the right hosting solution for your project or client's dynamic web application/services requirements.**

***OUTSOURCING***:

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

***J2EE HOSTING***:

Below is a partial price list of our standard hosting plans. (**Reseller accounts also available**). For more details please log on to **http://www.webappcabaret.com/jdj.jsp**

| Features | Basic | Professional | Enterprise | Dedicated |
|---|---|---|---|---|
| Monthly Cost | us$9.00 | us$17.00 | us$39.00 | us$191.00 |
| Servlet Spec | Latest | Latest | Latest | Latest |
| JSP Spec | Latest | Latest | Latest | Latest |
| EJB Spec | | | Latest | Latest |
| Private JVM | x | x | x | x |
| Database | 1 | 1 | 5 | Unlimited |
| e-Commerce | | x | x | x |
| Tier 1 Center | x | x | x | x |
| Accounts/Server | 120 | 120 | 35 | 1 |
| RAM/Server | 2GB | 2GB | 2GB | 256MB |
| Max Memory Heap | 20 MB | 30 MB | 60 MB | 256MB |
| RAID Diskspace | 60MB | 200MB | 900MB | 40GB |
| Bandwidth/Month | 3GB | 10GB | 20GB | 55GB |
| Backup | x | x | x | x |
| Domains | 1 | 2 | 5 | Unlimited |
| Email Accounts | 2 | 20 | 100 | Unlimited |
| Servlet Contexts | 1 | 4 | 20 | Unlimited |
| Control Panel | x | x | x | x |
| FTP | x | x | x | x |
| Telnet/SSH | | x | x | x |
| Web Mail | x | x | x | x |
| Web Stats | | x | x | x |
| Perl/PHP | | x | x + ASP.NET | x + ASP.NET |
| Dedicated IP | | | x | x |
| Engine Choices | x | x | x | x |
| WAR/EAR | x | x | x | x |
| Tomcat | | Latest | Latest | Latest |
| JBoss | | | Latest | Latest |
| Jetty | Latest | Latest | Latest | Latest |

Prices, Plans, and Terms subject to change without notice. Please log on to our website for the latest Prices and Terms.

Stream might be a better choice if you're going to be recording lengthy samples and memory becomes scarce. Of course, recording the sample is just half of the story. Now we have to play it back.

### *Playing Back Audio*

Playing back a previously recorded audio track is essentially the reverse of recording it. That is, the sample's bytes, originally stored in an OutputStream, are written out to a SourceDataLine one chunk at a time until there's nothing left or until the user clicks stop.

To read the bytes a chunk at a time, we'll need an InputStream. The Java Sound API provides the class AudioInputStream that has several convenience methods for working with samples. Again, we'll need to refer to the same AudioFormat that the sample was originally recorded in. In our case, we'll assume we're dealing with a completely in-memory sample, expressed as an array of bytes (see Listing 9).

Note that AudioInputStream's mark method is used to mark the beginning of the sample, while the reset method is used to "rewind" the sample to the beginning.

As has been the case, we'll need a separate thread to play back the sample. We'll use the AudioInputStream set up above to read sample bytes from it, a chunk at a time, writing them out to a SourceDataLine. Just as we obtained our TargetDataLine from the AudioSystem, we'll obtain a SourceDataLine suitable for playing back a sample in our AudioFormat through inquiry (see Listing 10).

Since we have a SourceDataLine that can handle our AudioFormat, we can start a thread to write out the sample bytes to it (see Listing 11).

Now that you have your audio track playing back – we're almost done!

## Putting It All Together

At this point we have the main ingredients for a basic multi-track MIDI sequencer that can also record and play back audio. Although we can play back multiple tracks of MIDI using just one thread, it's much more difficult to play back multiple samples with a single thread. For simplicity, we'll continue to use one thread for all MIDI data, but create a different thread for each audio sample.

The basic trick for integrating MIDI and one or more samples is to simply synchronize the start of the MIDI tracks thread with the audio track thread(s) using normal thread sychronization techniques.

Of course, real commercial MIDI/audio sequencers can do much more than record and play back multiple tracks. That's just the beginning. After all, a real sequencer can:

- Play back what was recorded at one tempo at a different tempo
- Import "instrument definitions" that specify the patch names mapped to patch numbers
- Select each track's "patch" by searching the available patches by name
- Provide a mixer with volume and pan sliders for each track
- Record and play back volume changes from the mixer in real time
- "Trigger" audio samples from the keyboard (a la a conventional sampler)
- Quantize recorded MIDI data to the nearest 1/4 note, 1/8th note, 1/16th note, etc.

I'm out of space, so for now, I'll have to leave that as an exercise for you, the reader. In the meantime, enjoy your new sequencer!

**Mike Gorman** is a senior software architect for J.D. Edwards, a PeopleSoft company, concentrating on J2EE distributed transaction systems. Mike has been coding in Java since 1997. In his spare time, Mike plays with MIDI, Swing, Web services, and JDO.

*mike_gorman@jdedwards.com*

## References

- *Open source MIDI and audio projects: Audio Development System:* http://sourceforge.net/projects/adsystem
- *jMusic:* http://sourceforge.net/projects/jmusic
- *Sound Grid:* http://sourceforge.net/projects/soundgrid

## API References

- *Java Sound Programmer Guide:* http://java.sun.com/j2se/1.4.1/docs/guide/sound/programmer_guide/contents.html
- *Java Sound Demo:* http://java.sun.com/products/java-media/sound/samples/JavaSoundDemo/

## MIDI Specification

- *Official MIDI Specification:* www.midi.org
- *Online MIDI Specification (unofficial):* www.borg.com/~jglatt/tech/midispec.htm

## Miscellaneous

- *Bug ID 4773012: RFE: Implement a new stand-alone sequencer:* http://developer.java.sun.com/developer/bugParade/bugs/4773012.html

- *Bug ID 4783745: Sequencer cannot access external MIDI devices:* http://developer.java.sun.com/developer/bugParade/bugs/4783745.html

**Listing 1:** Displaying the MIDI devices available

```
MidiDevice.Info[] info =
 MidiSystem.getMidiDeviceInfo();

 for (int i=0; i < info.length; i++) {
 log(i + ") " + info[i]);
 log("Name: " + info[i].getName());
 log("Description: " +
   info[i].getDescription());

  MidiDevice device =
    MidiSystem.getMidiDevice(info[i]);
  log("Device: " + device);
}
```

**Listing 2:** Sending "middle C note on" and "note off"

```
// For each MidiDevice, open it up,
// obtain it's receiver, and try it out
MidiDevice dev = getDevice();
dev.open(); //(at program start)
Receiver receiver = dev.getReciever();

// Send middle C (60) "note on"
// at maximum velocity (127)
ShortMessage msg1 = new ShortMessage();
msg1.setMessage(ShortMessage.NOTE_ON,
  60, 127);
receiver.send(msg1, -1);

// Wait a second
Thread.sleep(1000);

// Send middle C "note off"
ShortMessage msg2 = new ShortMessage();
msg2.setMessage(ShortMessage.NOTE_OFF,
  60, 0);
receiver.send(msg2, -1);

// Close the device (at program exit)
dev.close();
```

**Listing 3:** Minimal Receiver implementation

```
public class MyReceiver extends Object
  implements Receiver {
  public void send(MidiMessage msg,
    long time) {
    log(“Received message “ + msg);
  }

  public void close() {
    log(“Closing”);
  }
}
```

J2ME | J2SE | J2EE | Home

**Listing 4:** Listen to each device's transmitter

```
// Listen for MIDI messages originating
// from each MidiDevice
MidiDevice device = getDevice();
device.open(); // (at program start)

// Hook up a receiver to the transmitter
device.getTransmitter().setReceiver(
  new MyReceiver());

// Wait long enough to play a few notes
// on the keyboard
Thread.sleep(30000);

// Close the device (at program exit)
device.close();
```

**Listing 5:** Sample metronome

```
public class Metronome extends Object
  implements Runnable {
  private Receiver _receiver;
  private ShortMessage _accentOn;
  private ShortMessage _accentOff;
  private ShortMessage _nonAccentOn;
  private ShortMessage _nonAccentOff;
  private boolean _stopped = true;

  public Metronome(MidiDevice rcvDev) {
    super();
    _receiver = rcvDev.getReceiver();
    _accentOn = createNoteOnMsg(42,127);
    _accentOff = createNoteOffMsg(42);
    _nonAccentOn = createNoteOnMsg(42,90);
    _nonAccentOff = createNoteOffMsg(42);
  }

  private ShortMessage createNoteOnMsg(
    int note, int velocity) {
    ShortMessage msg = new ShortMessage();
    msg.setMessage(ShortMessage.NOTE_ON,
      note, velocity);
    return msg;
  }

  private ShortMessage createNoteOffMsg(
    int note) {
    ShortMessage msg = new ShortMessage();
    msg.setMessage(ShortMessage.NOTE_OFF,
      note, 0);
    return msg;
  }

  public void startMetronome() {
    _stopped = false;
    new Thread(this).start();
  }

  public void stopMetronome() {
    _stopped = true;
  }

  public void run() {
    long startTime =
      System.currentTimeMillis();
    try {
      while (_stopped == false) {
        _receiver.send(_accentOn, -1);

        Thread.sleep(100);
        _receiver.send(_accentOff, -1);
        Thread.sleep(
          getTimeTillNextBeat(startTime));
        for (int i=0; i < 3; i++) {
          _receiver.send(_nonAccentOn,
            -1);
          Thread.sleep(100);
          _receiver.send(_nonAccentOff,
            -1);
        Thread.sleep(getTimeTillNextBeat(
          startTime));
         }
       }
    } catch (InterruptedException e) {
      e.printStackTrace();
      _stopped = true;
    }
  }

  // assumes 120 bpm (or 500ms per beat)
    private static long getTimeTillNextBeat(
    long startTime) {
    long position =
      System.currentTimeMillis() –
      startTime;
    long timeRemaining = position % 500;
    return timeRemaining;
  }
}
```

**Listing 6:** Rebroadcasting incoming MIDI messages on the desired MIDI channel

```
public void send(MidiMessage msg,
  long time) {
  try {
    if (msg instanceof ShortMessage) {
      // Play back the incoming msg on
      // the desired channel
      ShortMessage incomingMsg =
        (ShortMessage) msg;
      ShortMessage playbackMsg =
        new ShortMessage();

      // Change the incoming message
      playbackMsg.setMessage(
        incomingMsg.getCommand(),
        _playbackChannel,
        incomingMsg.getData1(),
      incomingMsg.getData2());
      _receiver.send(playbackMsg, -1);

      // If the sequencer is currently
      // recording, hold on to each msg
      if (_recordEvents) {
        // Take note of when the first
        // msg arrives so we'll know
        // when to start playback later
        if (_firstMessageArrivedAt == 0) {
          _firstMessageArrivedAt =
            System.currentTimeMillis();
         }
        _recordedEvents.addElement(
          new MyEvent(playbackMsg, time));
      }
    }
  } catch (InvalidMidiDataException e) {
    e.printStackTrace();
  }
}
```

**Listing 7:** Preparing to record audio

```
DataLine.Info info = new DataLine.Info(
  TargetDataLine.class, getAudioFormat());

if (AudioSystem.isLineSupported(info) ==
  false) {
  log("Line matching " + info +
    " not supported.");
  return;
}

TargetDataLine targetLine =
  (TargetDataLine) AudioSystem.getLine(
  info);
targetLine.open(getAudioFormat(),
  targetLine.getBufferSize());

// Create an in-memory output stream and
// initial buffer to hold our samples
ByteArrayOutputStream baos =
  new ByteArrayOutputStream();
int frameSizeInBytes =
  getAudioFormat().getFrameSize();
int bufferLengthInFrames =
  targetLine.getBufferSize() / 8;
int bufferLengthInBytes =
  bufferLengthInFrames * frameSizeInBytes;
byte[] data = new byte[bufferLengthInBytes];
```

**Listing 8:** Recording audio

```
public void run() {
  getTargetLine().start();

  while (isRecording()) {
    int numBytesRead =
      getTargetLine().read(getData(), 0,
      getBufferLengthInBytes());
    if (numBytesRead == -1) {
      break;
    }
    getOutputStream().write(getData(), 0,
      numBytesRead);
  }
  getTargetLine().stop();

  // flush and close the output stream
  try {
    getOutputStream().flush();
    getOutputStream().close();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

**Listing 9:** Preparing for audio playback

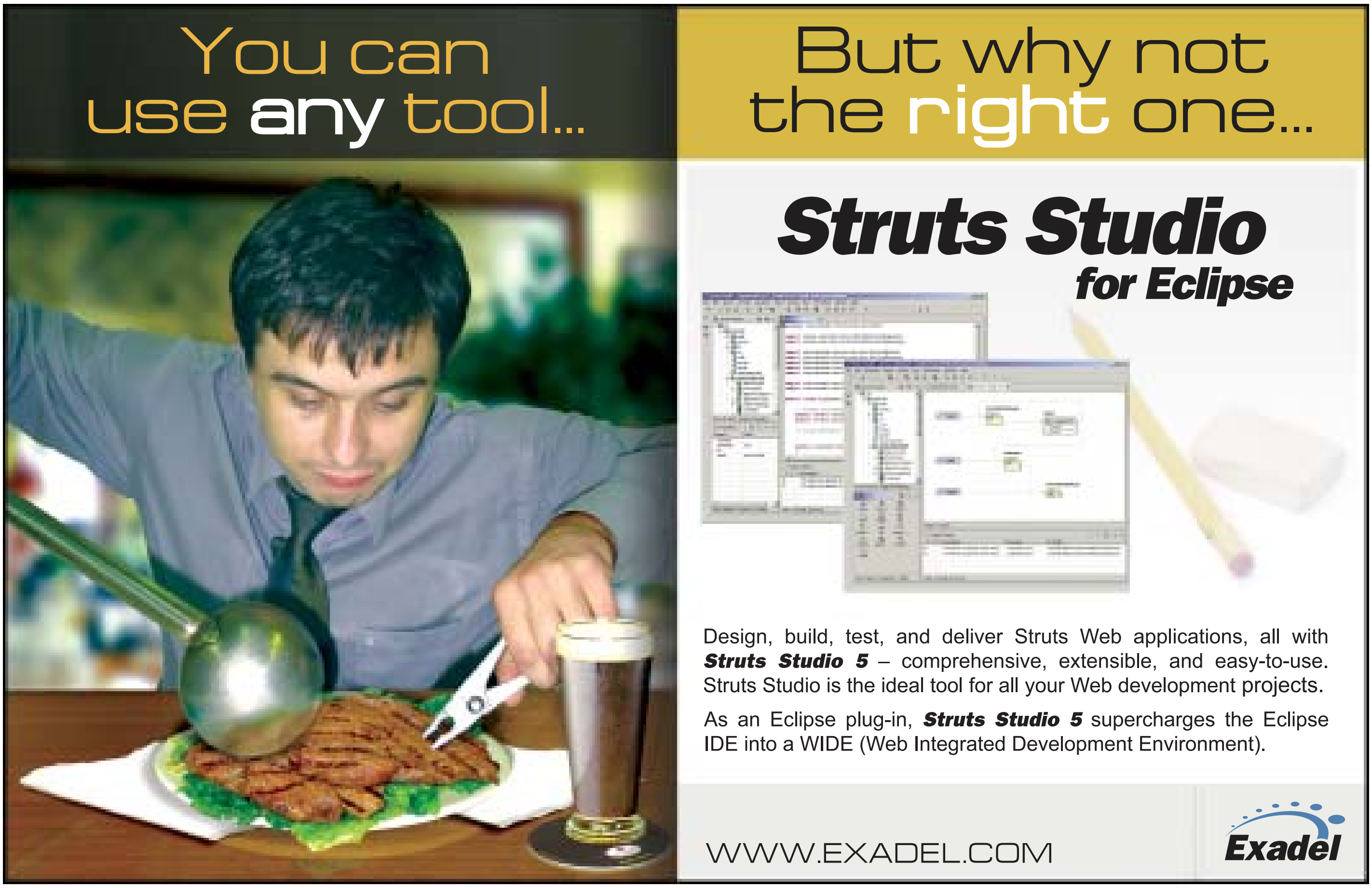
```
byte[] data = getSampleBytes();

int frameSizeInBytes =
  getAudioFormat().getFrameSize();
AudioInputStream audioInputStream =
  new AudioInputStream(
new ByteArrayInputStream(data),
  getAudioFormat(), data.length /
  frameSizeInBytes);

try {
  audioInputStream.mark(2000000000);
  audioInputStream.reset();
} catch (IOException e) {
  e.printStackTrace();
  return;
}

long duration = (long)
  ((audioInputStream.getFrameLength() *
  1000) / getAudioFormat().getFrameRate());
```

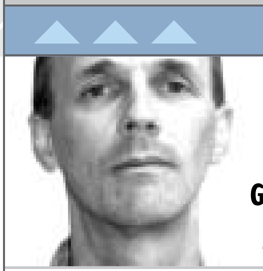
**Listing 10:** Initializing a SourceDataLine

```
// Define the required attributes for
// our line, and make sure a compatible
// line is supported.
DataLine.Info dlInfo = new DataLine.Info(
  SourceDataLine.class, getAudioFormat());
if (AudioSystem.isLineSupported(dlInfo)
  == false) {
  throw new Exception("Line matching " +
    dlInfo + " not supported.");
}

getAudioInputStream().reset();

// Get and open the source data line for
// playback.
SourceDataLine sourceLine =
  (SourceDataLine) AudioSystem.getLine(
  dlInfo);
int bufSize = 16384;
sourceLine.open(getAudioFormat(),
  bufSize);
```

**Listing 11:** Playing back audio

```
public synchronized void run() {
  try {
    // play back the captured audio data
    int frameSizeInBytes =
      getAudioFormat().getFrameSize();
    int bufferLengthInFrames =
      getSourceLine().getBufferSize() / 8
    int bufferLengthInBytes =
      bufferLengthInFrames *
      frameSizeInBytes;
    byte[] data = new byte[
      bufferLengthInBytes];

    // start the source data line
    sourceLine.start();

    // main playback loop
    while (isPlaying()) {
      // rewind at start of each loop
      getAudioInputStream().reset();
      while (true) {
        int numBytesRead =
          getAudioInputStream().read(
          data);

        if (numBytesRead == -1 ||
          isPlaying() == false) {
          break;
        }

        int numBytesRemaining =
          numBytesRead;

        while (numBytesRemaining > 0) {
          numBytesRemaining -=
            sourceLine.write(data, 0,
            numBytesRemaining);
        }
      }

      // We've reached the end of the
      // stream.  Let the data play out,
      // then stop and close the line.
      sourceLine.drain();
    }
    sourceLine.stop();
    sourceLine.close();
  } catch (LineUnavailableException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  } catch (InterruptedException e) {
    e.printStackTrace();
  } catch (JStudioException e) {
    e.printStackTrace();
  }
}
```





**Glen Cordrey**
J2ME Editor

# Hanged in a Fortnight?

Samuel Johnson said, "When a man knows he is to be hanged in a fortnight, it concentrates his mind wonderfully." While Sun's current situation may not be dire enough to be considered analogous to facing the hangman's rope, it is clear that economic distress is forcing Sun to change its mindset. Whether that change is a concentrating focus or a casting about for a lifeline is subject to debate.

Now I won't speculate about the likelihood of Sun being acquired by IBM, HP, Dell, or anyone else, other than to say that Scott McNealy's egotism plus Sun's $5.7 billion cash reserve would certainly make such a takeover attempt entertaining. Taking away a favorite toy from a petulant child can be difficult enough, but if that child can afford a phalanx of lawyers and other corporate hatchet men, the attempt can rival the best overblown soap opera on TV.

I will, however, remark upon recent events.

The telecommunications industry was hit particularly hard by the bursting (aka the dot bomb) of the tech bubble, and the telecom industry was an important buyer of Sun's servers. Sun's revenue stream from hardware also continues to shrink as a result of a corporate shift to cheaper Intel boxes and Linux, so Sun is placing greater marketing emphasis on software.

At this year's JavaOne, Sun launched a campaign to increase public recognition of the Java brand name. "Java Powered" is to become the catchphrase for this campaign, which includes a redesigned Java cup (with bolder strokes) and redesigned Web sites (java.net for developers, java.com for the public). Sun and its Java partners have unleashed a $500 million advertising onslaught to promote "Java Powered" with Christina Aguilera as its eye candy. All well and good since, as I wrote in a previous editorial, I felt that Sun was relying too heavily on its technology laurels and not being effective in marketing to the general public (I used the success of the WiFi branding as an example of the desired goal). Although I do think this Christina business may have gone too far with the "Christina Everywhere" J2ME application touted on Sun's page (www.java.com/en/explore/ mobile/christina.jsp) and available from Nextel. Not exactly what I had in mind when writing MIDlets for Nextel, but then maybe that explains why I haven't retired on my MIDlet licensing fees.

Now comes the Java Enterprise System (née Project Orion) and the Java Desktop System, and I'm perplexed. First realize that neither one is just Java; rather, multiple technologies are bundled under the Java moniker, presumably because Java has the widest public recognition. Of course, this may only matter to us techies because we know what Java really is, but I can't help but wonder whether imprecision here reflects a lack of cohesion in the overall effort.

One rationale for Orion, as stated at www.sun.com/2003-0930/feature/, is that "customers are tired of…acquiring enterprise infrastructure software from multiple vendors." Am I the only one who sees some irony in this statement from a company whose "write once, run anywhere" and "the network is the computer" philosophies embrace heterogeneous systems? Isn't the hegemony of a single-source solution what we're trying to avoid vis-à-vis Microsoft?

The Java Desktop System is a pseudo-MS Windows and Office without the MS. While I admire Sun's dogged attempt to provide an alternative to MS, what does it have to do with Java? Is there a realistic chance of wresting the desktop from Microsoft, or is this largely fueled by McNealy's antagonism toward all things Bill?

Now I'm not expecting Sun to be eclipsed – they still have superior technology, and they're pointed in the right direction (in my humble opinion) in branding and promoting Java technology. In the J2ME arena, in particular, they have an especially strong position, as evidenced by the continued increase in the number of hardware platforms and network providers supporting J2ME. A revival of the tech economy and, in particular, the telecommunications industry, could make all other factors largely irrelevant by increasing the demand for both network servers and enterprise (J2EE) and mobile (J2ME) software. I just wish Sun's marketing strategy seemed more coherent to me. But then I'm an engineer, and I've learned that marketing employs a different logic than the logic I understand.

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

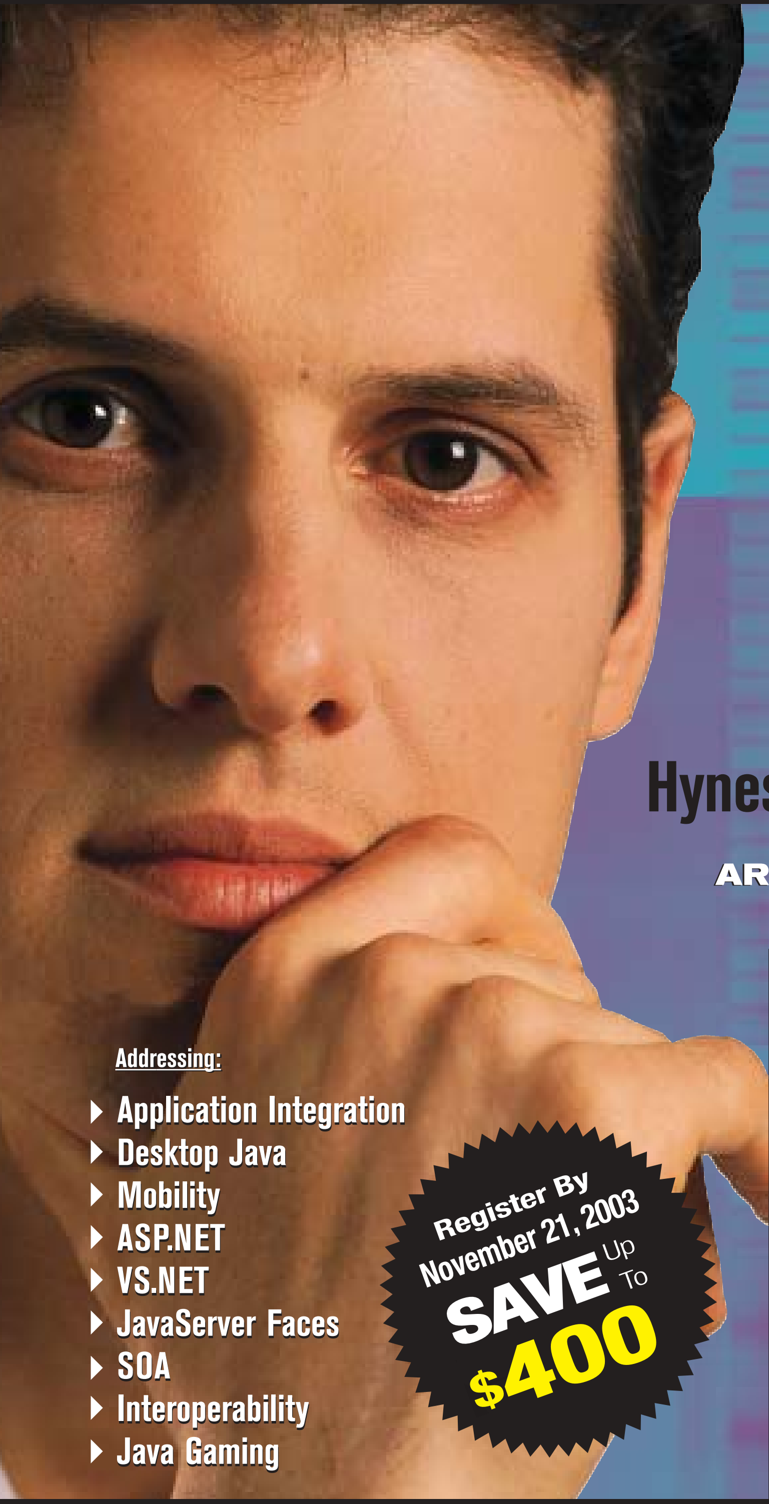
*glencordrey@sys-con.com*

**Hanged in a Fortnight?**

Samuel Johnson said, "When a man knows he is to be hanged in a fortnight, it concentrates his mind wonderfully." While Sun's current situation may not be dire enough to be considered analogous to facing the hangman's rope, it is clear that economic distress is forcing Sun to change its mindset.

52

# International Conference & Expo



# Edge 2004 EAST

## Development Technologies Exchange

## February 24-26, 2004

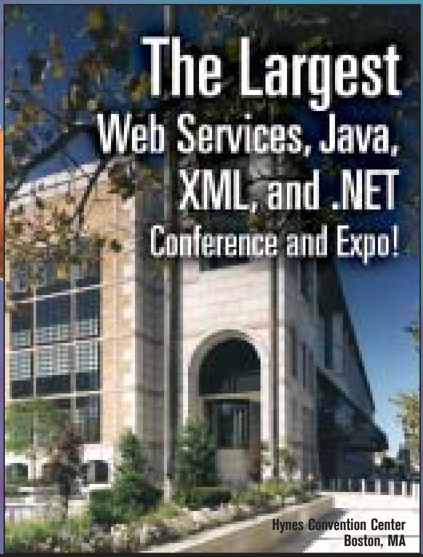
## Hynes Convention Center, Boston, MA

ARCHITECTING JAVA, .NET, WEB SERVICES, OPEN SOURCE, AND XML

Addressing:

- Application Integration
- Desktop Java
- Mobility
- ASP.NET
- VS.NET
- JavaServer Faces
- SOA
- Interoperability
- Java Gaming

Register By November 21, 2003 SAVE Up To $400



The Largest Web Services, Java, XML, and .NET Conference and Expo!

Hynes Convention Center
Boston, MA

For more information visit
**www.sys-con.com**
or call
**201 802-3069**

Over 200 participating companies will display and demonstrate over 500 developer products and solutions.

Over 3,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 100 of the latest sessions on training, certifications, seminars, case-studies, and panel discussions promise to deliver real world benefits, the industry pulse and proven strategies.

**Full Day Tutorials Targeting**

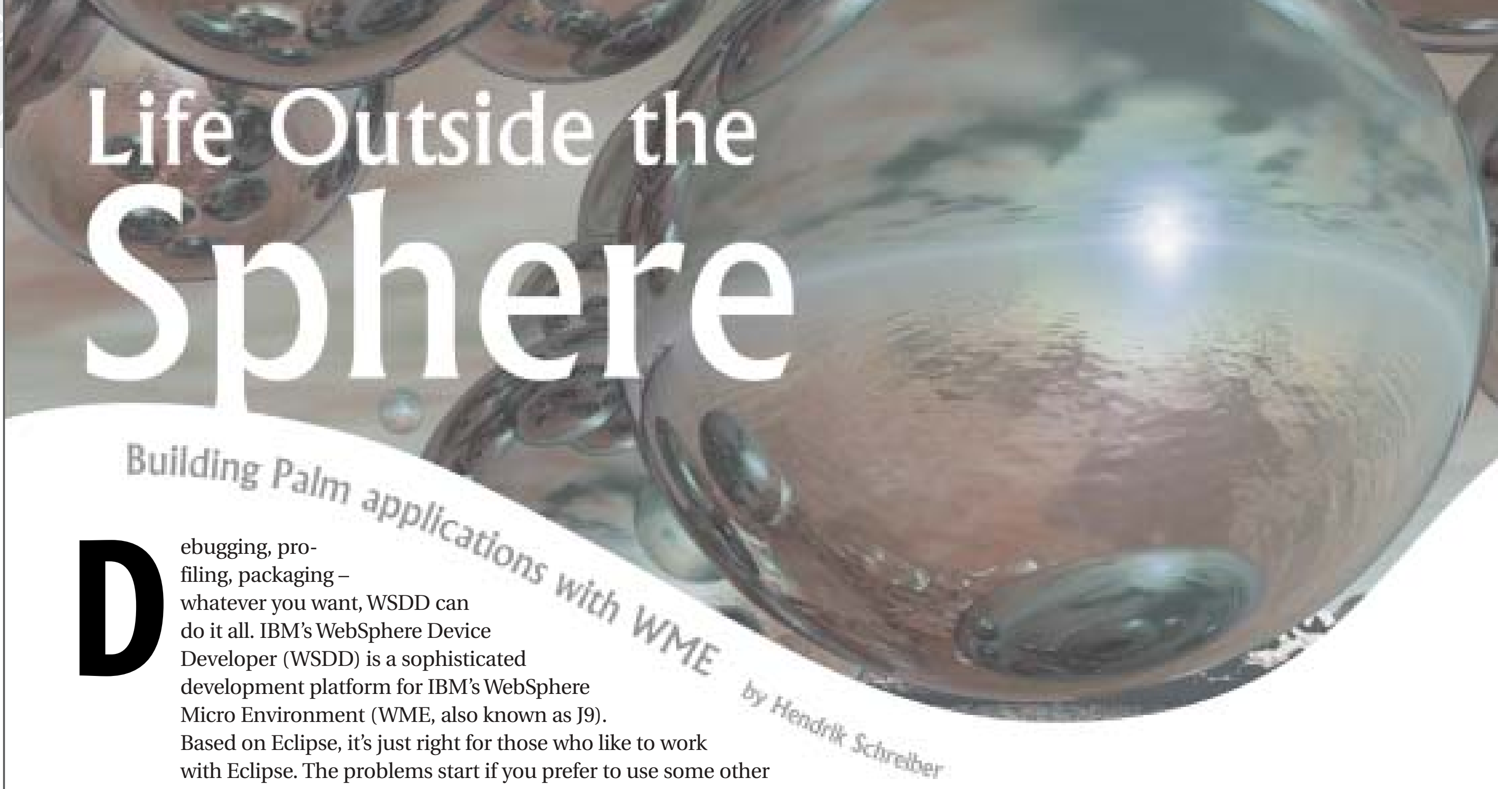
• Java • .NET • XML
• Web Services • Open Source

Conference program available online!
www.sys-con.com/edgeeast2004

**Contact information: 201 802-3069 • events@sys-con.com**

OWNED BY SYS-CON MEDIA

PRODUCED BY SYS-CON EVENTS



# Life Outside the Sphere

## Building Palm applications with WME

by Hendrik Schreiber

Debugging, profiling, packaging – whatever you want, WSDD can do it all. IBM's WebSphere Device Developer (WSDD) is a sophisticated development platform for IBM's WebSphere Micro Environment (WME, also known as J9). Based on Eclipse, it's just right for those who like to work with Eclipse. The problems start if you prefer to use some other IDE or you believe in automated, continuous integration. This article will show you how to master using WME without WSDD.

WSDD uses Ant build scripts, but effectively hides the implementation of its special tasks for the SmartLinker jxelink and other tools. If you want to build a WSDD project outside of WSDD, you can't rely on automatically generated Ant build files. This makes it hard to build a project from the command line and therefore rules out automatic builds. The necessary tasks are simply not accessible. On top of projects not being exportable, it's fairly difficult for inexperienced users to import an existing project into WSDD.

With this said, why would you use WSDD if it locks you in? Well, as mentioned before, it has a couple of nice features and – this is probably the main reason – if you want to use WME you have to install WSDD. IBM unfortunately does not offer WME without WSDD. Also since Big Blue seems to concentrate its documentation efforts on the IDE rather than the VM, it's only natural to use the IDE for convenience.

Luckily all the WSDD's custom Ant tasks are included as command-line tools in the WME. This allows us to call them using the <exec> task. Admittedly, there is a bit of irony here, because some of these tools were originally written in Java.

### Palm as an Example

Let's look at an example that shows how to build a deployable application for J9 on Palm using Ant. To reference the base of the J9 installation, it makes sense to define a corresponding property. The J9 folder is usually installed in a subfolder of IBM/Device Developer/wsddx.y called ive, therefore we name the property ive. For the etymologists among you: ive is short for "IBM VisualAge Embedded," short for "IBM VisualAge Embedded Systems, Java Edition," which is the original name for VAME or "VisualAge Micro Edition," WSDD's successful predecessor (winner of the **2002 *JDJ* Reader's Choice Award** for Best J2ME IDE).

The first step in producing our application is to compile the Java classes. IBM recommends using the J9 compiler j9c. Like all other J9 tools, j9c can be found in the folder ${ive}/bin. Experience shows that you're probably equally well off with a Sun compiler. To compile for a specific profile you have to include its classes in the bootclasspath. Typically you'll find the appropriate classes in ${ive} /lib/jcl<yourProfile>/classes.zip. Don't ask me why IBM doesn't stick to the convention of using JARs instead of zips. However, equipped like this, compiling should be a piece of cake.

```
<javac
   srcdir="src"
   destdir="classes"
   bootclasspath=
   "${ive}/lib/jclCldc/classes.zip"
/>
```

Once you have compiled all the sources, package them into the J9 archive format JXE (Java eXEcutable). This is done with the jxelink SmartLinker – a tool that not only packages, but also completely rearranges the bytecode. For platforms other than Palm and QNX it can even compile bytecode into native code (ahead-of-time compilation, AOT). Because of space constraints I won't delve into the many options of this tool; I'll only describe how to call it from Ant. Also, because there are so many options, it makes sense to use an extra file for them, just like WSDD does. The location of this file can be passed as an argument to the jxelink executable with a prefixed @. To access properties defined in the Ant file, you can declare macros for the link option file, for example:

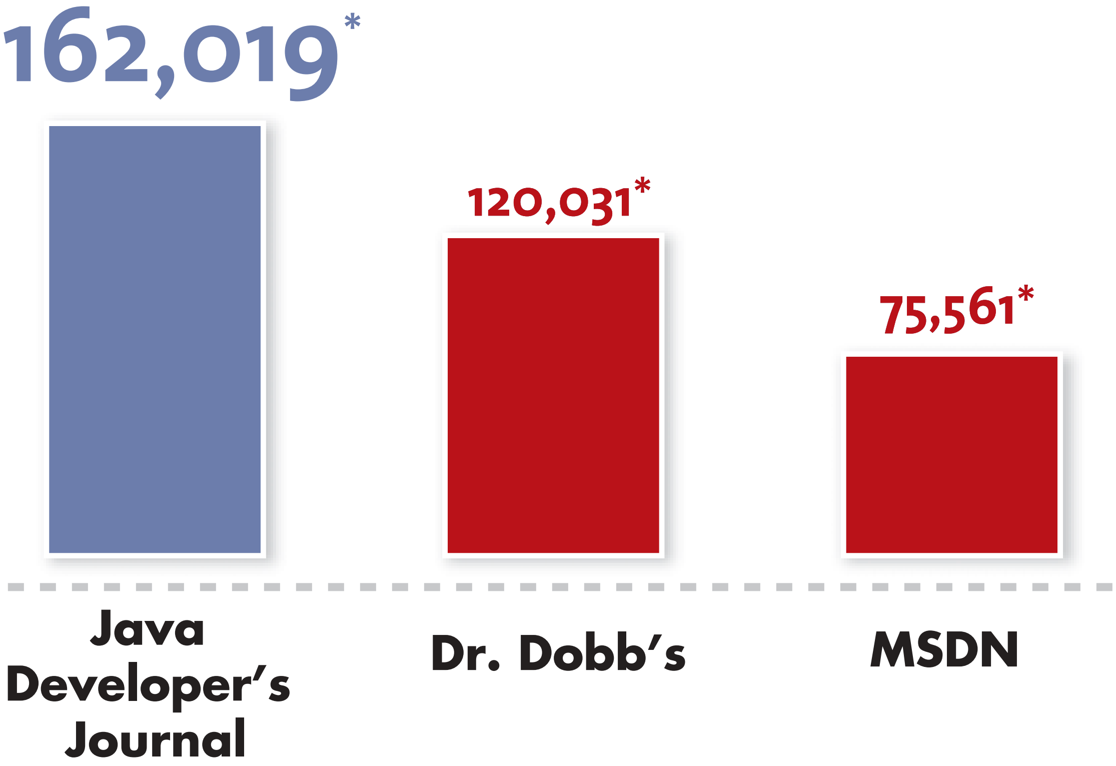
```
<exec
executable="${ive}/bin/jxelink.exe">
   <arg value="-macro"/>
   <arg value="BASEDIR=${basedir}"/>
   <arg value="-o"/>
   <arg file="jxe/MyApp"/>
   <arg value="@${basedir}/palm.link"/>
</exec>
```

This code sets the macro BASEDIR to the same value as the property ${basedir} so that we can pass it into the options file ${basedir}/palm.link, then we can reference it with double curly parentheses like this {{BASEDIR}}. For example, if you put the line "-cp "{{BASEDIR}}/SomeLib.jar"" into the link options file, ${basedir}/SomeLib.jar will be added to the SmartLinker's classpath.

As the maximum segment size for the Palm is 64KB, jxelink has to be configured to produce multiple JXE files for applica-

# #1 Circulation in the World!

*JDJ is the highest circulation i-Technology magazine in the world!*



*JUNE 2003 BPA AUDIT STATEMENTS *All circulation numbers are publishers' most current own data, six-month average circulation through June 2003.

SYS-CON MEDIA

Miles Silverman
VP Marketing

Carmen Gonzalez
Senior VP Marketing

**Contact** | Carmen Gonzalez: carmen@sys-con.com or 201 802.3021

JAVA DEVELOPER'S JOURNAL

tions exceeding this limit. Therefore it usually makes sense to specify a separate output directory. In the previous code snippet we achieve this with the –o option. As a side note, this also means that each of your compiled classes must not be larger than 64KB, which can be tricky when you're using large arrays.

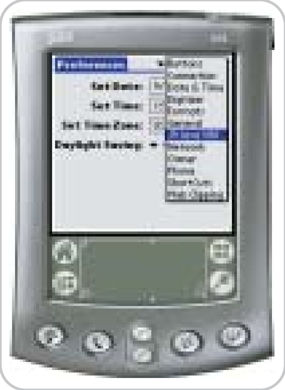
Once you have produced the necessary JXE files, you can proceed to build Palm resource files. If you are familiar with Palm development, you've probably already used the Palm resource compiler PilRC. This free tool compiles GUI definitions into binary resources, which you can use from your code. This makes sense, particularly when you're using the nongraphical CLDC (Connected Limited Device Configuration) because you want to reduce the footprint or escape the MIDP sandbox, but still need to use a GUI.

Like other VMs, J9 comes with wrapper classes for PalmOS (located in ${ive}/ runtimes/ palmos/68k/ ive/lib/ palmos.zip) that let you call the needed OS functions for presenting a GUI defined with PilRC. As the wrapper is fairly thin, this unfortunately means that you have to manually allocate and free memory. To those of us who got to appreciate garbage collection, this is really nasty…. However, to compile our GUI resources we simply invoke PilRC with the <exec> task.

```
<exec executable="pilrc.exe">
   <arg file="MyApp.rcp"/>
   <arg path="bin"/>
</exec>
```

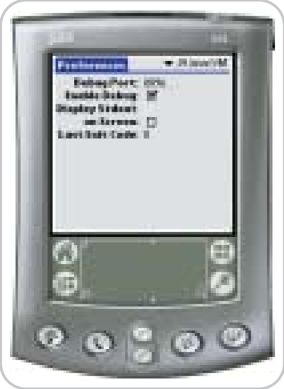
The first argument is the filename of the resource description file and the second is the output directory for the compiled results. Again, for brevity, I will not explain the PilRC file format as the compiler comes with a comprehensive and easy-to-understand manual.

Now we are getting to the final step: building the PRC (PalmOS Resource Collection) file. For this purpose we use J9's jxe2prc command-line tool. It takes all your compiled code and packages it into an executable PRC file. As we have to pass all the binary resource files on the command line, we need to build a corresponding property that contains all the filenames. We'll build this property by creating a <fileset> that contains all the binary files and then convert this file collection into a single property using the <pathconvert> task. Note that in order to avoid problems with spaces in the base directory, we substitute the base directory ${basedir} with a dot "." using <map>.



Preferences ▾ Buttons
Set Date:
Set Time:
Set Time Zone:
Daylight Saving: ▾

**Figure 1** Accessing the J9 preferences dialog



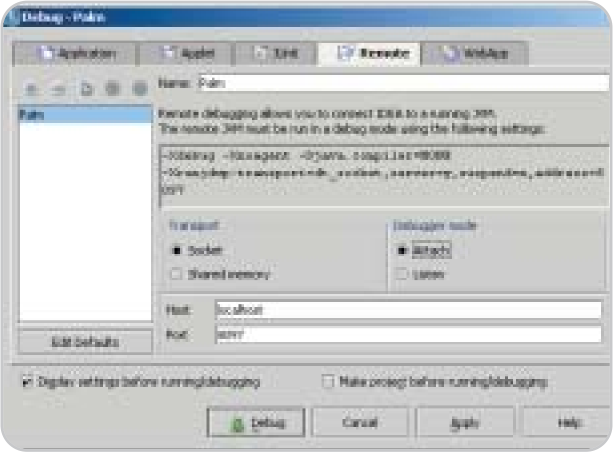
Preferences ▾ J9 Java VM
Debug Port: 8000
Enable Debug: ☑
Display Stdout on Screen: ☐
Last Exit Code: 0

**Figure 2** Checking the "enable debug"

```
<fileset dir="bin" id="bin.files.id">
   <include name="*.bin"/>
</fileset>
<pathconvert pathsep=" "
   property="bin.files"
   refid="bin.files.id">
      <map from="${basedir}" to="."/>
</pathconvert>
```

Now that we can reference the files generated by PilRC, we call jxe2prc with the appropriate arguments. These are (in this order):

- The four character Palm creator ID
- Your application name
- The main JXE file and all bin files we just generated
- Name of the output file



Debug - Palm

Name: Palm

Remote debugging allows you to connect IDEA to a running JVM.
The remote JVM must be run in a debug mode using the following settings:

```
-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8
097
```

Transport: ● Socket ○ Shared memory

Debugger mode: ● Attach ○ Listen

Host: localhost

Port: 8097

☑ Display settings before running/debugging ☐ Make project before running/debugging

**Figure 3** Remote debugging configuration for IntelliJ IDEA 3.0

```
<exec
   executable="${ive}/bin/jxe2prc.exe">
   <arg value="myid"/>
   <arg value="MyApp"/>
   <arg file="jxe/MyApp.jxe"/>
   <arg line="${bin.files}"/>
   <arg file="MyApp.prc"/>
</exec>
```

Done. With the help of the Palm install tool you can now synchronize the freshly built PRC file to your Palm and take your application for a spin – provided that you've already installed J9. This is done by synchronizing the PRC files found in ${ive}/runtimes/ palmos/68k/ive/ bin to your Palm. You'll need only one of the files, either midp20.prc or cldc20.prc, depending on which of the two you want to use.

Chances are you'll want to try your application out in the emulator before you actually test it on a real device. To conveniently start your application with Ant, first create a Palm emulator session file (psf) with J9 already installed. To create this session, just install the PRC files from ${ive}/runtimes/palmos/ 68k/ive/bin in a session with a clean Palm ROM and save the session – e.g., under j9cldc_run.psf (see sidebar on how to obtain a ROM file). Then start the emulator like this:

```
<exec executable="emulator.exe">
   <arg value="-psf"/>
   <arg file="j9cldc_run.psf"/>
   <arg value="-load_apps"/>
   <arg file="MyApp.prc"/>
   <arg value="-run_app"/>
   <arg value="MyApp"/>
   <arg value="-quit_on_exit"/>
</exec>
```

This task will automatically install and start your application. The -quit_on_exit option causes the emulator to automatically shut down once you exit your application. If you don't specify the -quit_on_exit option, it's crucial that you don't save your emulator session. Otherwise you won't have a clean environment the next time you start your application this way.

If you're like every other developer, something is probably buggy in your application. The emulator lets you write to STDERR and STDOUT, but System.err.println-debugging is a little backward and certainly a time-consuming matter. What you really need is a debugger. As WME supports the Java Debug Wire Protocol (JDWP), you can attach the debugger of your choice to J9. Just take a short detour.

## How to Obtain a Palm ROM File

There are two ways to obtain a Palm ROM file:

1. Become a member of the PalmSource (the software side of Palm) developer program and download ROMs from the program's Web site.
2. Transfer the ROM from a Palm device to your desktop machine using the ROM Transfer.prc file that comes with the Palm emulator. Directions are included in the emulator's manual.

Either way works equally well. However, to deliver your application to a wide audience with many different Palm devices, join the Palm developer program and test your application with as many different ROMs as possible.

## MIDP on Tungstens

The WME release included in WSDD 5.5 officially does not support Palm Tungstens. If you're just interested in deploying MIDP 1.0 applications on the Palm, look at the WME Toolkit for PalmOS developers, which supports Tungstens and is freely available from Palm. The toolkit lets you convert MIDP applications into PRC files for J9. Palm and IBM announced that future Tungsten devices will come with J9 preinstalled. Owners of Tungstens W, T2, and C who bought after October 1, automatically received a WME license bundled with their device.

At the time of writing IBM had only published a Technical Preview for their upcoming WSDD 5.6 release, which will also support Palm Tungsten devices.

## What's in an Acronym?

Confused by all the acronyms? You're not alone. Let's try to get it all straightened out.

- *Java Micro Edition (J2ME)*: Basically Java for small devices.
- *Connected Limited Device Configuration (CLDC)*: J2ME configuration for small devices.
- *Mobile Information Device Profile (MIDP)*: A profile based on CLDC, designed to extend and enhance the J2ME Connected Limited Device Configuration.
- *WebSphere Studio Device Developer (WSDD)*: An integrated development environment (IDE) for J2ME. Not to be confused with the Web site called WebSphere Developer Domain (WSDD).
- *WebSphere Micro Edition (WME)*: IBM's implementation of J2ME. This includes the JVM J9, originally written by the IBM subsidiary OTI.
- *Palm Resource Compiler (PilRC)*: A little freeware tool that compiles textual GUI definitions into binary resources.
- *Java Debug Wire Protocol (JDWP)*: An element of the Java Platform Debugger Architecture (JPDA) that defines how debugger and debuggee communicate with each other.
- *Palm OS Emulator (POSE)*: Free hardware emulator for Palm OS, published under GPL. Can only be used for Palm devices with Palm OS versions 5.0. The emulator mentioned in the article always refers to POSE, not the newer Palm OS Simulator.
- *Palm OS Simulator*: Palm OS 5.x recompiled for a desktop machine.
- *IBM VisualAge Embedded (IVE)*: Short for "IBM VisualAge Embedded Systems, Java Edition", which is the original name for "VisualAge Micro Edition" (VAME), WSDD's predecessor.

First you need to tell J9 that it should start in debugging mode. For this purpose, load the j9cldc_run.psf profile you created earlier, open the "Prefs" application, and select "J9 Java VM" (see Figure 1). Then check the "Enable Debug" checkbox (see Figure 2) and save the profile under the name j9cldc_debug.psf. This is now your debug base session.

J9 does not support JDWP directly. When linking with jxelink, all the debug symbols are stripped out of the JXE and put into a symbol file to minimize the JXE's size. Therefore your debugger needs to communicate with J9 through a tool called j9proxy. It takes the debuggee's and the debugger's addresses and the symbol file as arguments. The symbol file is located in the same directory as our JXE file and has the file extension sym.

All we have to do now is start the j9proxy, our application, and the debugger. We can easily start the proxy and the application in an Ant target.

```
<parallel>
<exec executable="emulator.exe">
   <arg value="-psf"/>
   <arg file="j9cldc_debug.psf"/>
   <arg value="-load_apps"/>
   <arg file="MyApp.prc"/>
   <arg value="-run_app"/>
   <arg value="MyApp"/>
   <arg value="-quit_on_exit"/>
</exec>
<exec
   executable="${ive}/bin/j9proxy.exe">
   <arg value="localhost:8096"/>
   <arg value="localhost:8097"/>
   <arg file="jxe/MyApp.sym"/>
</exec>
</parallel>
```

Note that we put the two <exec> tasks inside a <parallel> container task in order to start both the application and the proxy at the same time. With this setup we expect the Palm application to fulfill the JDWP server role, listening with a socket on port 8096. Our debugger needs to be configured to use a socket as a transport layer to attach to the proxy, which is listening on port 8097 for the debugger's connection. The proxy in turn connects to the application on the Palm (see Figure 3).

Once we've started the Ant target we just need to start our debugger and we're ready to roll. While debugging, resist using the many optimization options jxelink has to offer. Neither obfuscation nor inlining is very conducive to debugging – you may end up waiting for a long time before the application hits any breakpoints, simply because the line of code your breakpoint is referencing may not exist anymore.

There's no support for profiling J9 for Palm applications, so System.err-timestamps are your best bet. As the time inside the emulator is not the real time and usually goes by faster than on your PC, these timestamps are also a useful reality check about how slow your application would be on a real device.

## Conclusion

All in all IBM delivered a successful J2ME implementation, which can be integrated in a continuous integration software development process – thanks to the included command-line tools. Whether you like WSDD or not is a matter of taste. Anyhow, you can get around it. And if you run into problems on the way, just drop a line to the support newsgroup. Usually the folks from IBM/OTI answer quickly and with great expertise.

## Resources

- *WME:* www.ibm.com/software/wireless/wme/
- *WSDD:* www.ibm.com/software/wireless/ wsdd/
- *WSDD newsgroup:* news://news.software.ibm.com/ibm.software.websphere.studio.device-developer
- *PilRC :* www.ardiri.com/index.php? redir=palm& cat=pilrc
- *POSE :* www.palmos.com/dev/tech/tools/emulator/
- *Palm Simulator :* www.palmos.com/dev/tools/simulator/
- *JPDA/JDWP:* http://java.sun.com/products/jpda
- *MIDP WME Toolkit for PalmOS:* http://pluggedin.palm.com/regac/pluggedin/Toolkit
- *Project skeleton:* www.sys-con.com/java/sourcec. cfm

**Hendrik Schreiber** develops data synchronization solutions utilizing SyncML and J2ME/J2EE for Nexthaus in Raleigh, North Carolina. He is also co-author and author of two German Java-related books, published by Addison-Wesley.

*hs@tagtraum.com*

# The State of Web Services, 2003 A.D.

They're 'a tool for the times,' say the experts

by Jeremy Geelan

What do you get if you cross an early 21st-century visionary CTO with a late 19th-century employee of the Edison Electric Light Company? Answer: a fantastic keynote address at Web Services Edge 2003 West, held in Santa Clara last month.

The visionary in question was Allan Vermeulen, coauthor of the codehead's classic *The Elements of Java Style,* and now CTO of the world's largest online retailer, Amazon.com. The Edison employee was Sam Unsell, whose contribution to the development of technology – Vermeulen explained – was to develop an economic model for electricity use in Chicago.

As with electricity then, so with Web services now. This, in Vermeulen's view, is the next shoe that needs to drop.

"Somebody has to be the Sam Unsell of Web services," he proclaimed, meaning that someone in the Web services space has to come up with a good idea for what kind of economic model is best suited to underpinning the technology.

Commercially available electricity, he explained, was only able to catch on and become pervasive because, with Unsell's help, the Edison Electric Light Company invented not just the first commercially practical incandescent lamp but a complete electrical distribution system for light and power – including generators, motors, light sockets with the Edison base, junction boxes, safety fuses, underground conductors, and other devices.

The comparison held the packed audience at the Santa Clara Convention Center, quite literally, spellbound. It was deemed by all who attended to be one of the most memorable and – pun intended – illuminating keynotes in the history of the Web Services Edge series of Conferences and Expos, which is saying something since in previous years keynotes have been given by folks like the "Father of Java," Sun's James Gosling; and the "Father of Markup," Charles F. Goldfarb.

Vermeulen's ebullient opening keynote characterized well a conference that for three days brimmed with good content and animated discussions.

## The Complexity Crisis

Keynote discussion panels featured the likes of John Schmidt, CTO of the No. 1 specialty retailer in the U.S., Best Buy, who brought to bear his enormous real-world experience of Web services: Best Buy moves about 100 gigabytes of data a day – inventory data, foundation data (pricing, etc.) – and top management throughout industry, Schmidt reported, is starting to recognize the issues of complexity in IT.

"We need," he observed, "to help take layers of complexity out of our IT environment." Whereas Web services, in Schmidt's view, may take us in the opposite direction.

Coming from a seasoned expert like Schmidt, who also chairs the Methodology Committee of the EAI Consortium, this was a compelling message – especially once he had set the stage with a reference to what he called "the dark side of systems integration – the complexity crisis."

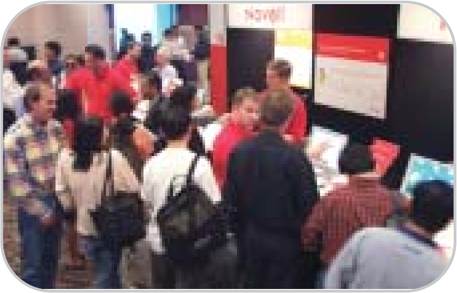
Best Buy alone has over 600 technologies to support 165 technology capabilities, Schmidt reported. "A couple of years ago it took about 20–30 days to build a complete interface," he said. "Nowadays it takes about 4–5 days. Best Buy now adds over 550 interfaces every month (over the past 3 months)."

In other words, and this was Schmidt's point, "As complex as our environment is at the moment, Web services is going to make it even more complicated."

A Web service can be built almost at the push of a button, Schmidt concluded. "Accordingly, they will proliferate on a massive scale."

## Keynote Panel: Web Services Paradigm Has Evolved

At another keynote discussion panel the question was "Interoperability: Is Web Services Delivering?"



“Do think of the ‘W’ in Web service as a way to ask ‘Why not?’ when presented with the difficulties or challenges of opening up a system or sharing information across departmental systems?”

**–Velan Thillairajah**

*EAI Technologies*

*Founding Member of the EAI Industry Consortium*

When panel moderator Derek Ferguson, editor-in-chief of *.NET Developer’s Journal,* asked the panel members to set the parameters of the discussion by first defining Web services, it became clear that the invited experts on the keynote stage agreed that, while defined by the interop protocol known as SOAP 1.1, no longer do Web services necessarily have to be XML, or even over HTTP. The paradigm has evolved.

David Chappell, VP and chief technology evangelist, Sonic Software, stressed that in his view, while Web services interactions do not have to be across HTTP, “XML is key to defining what a Web services interaction should be. It’s best suited for the role of serving as the language for describing the data that needs to be exchanged between applications.”

Gary Brunell, VP of professional services for Parasoft, pointed out that “If we’re going to use the term ‘Web services,’ it does suggest the Web, and so HTTP and HTTPS. XML is very important too,” he added.

Meantime, David White of Microsoft said he disagreed with the “Web” part of the term “Web services.” “I’m a big believer in transport agnosticism,” White said. “I’m really more concerned about the data representation and the invocation, rather than the transport. The key is to get something back and forth without great expense.”

Chappell agreed: “To me the ‘services’ word is the more important, the service-oriented architecture part. ‘Web services’ is now a more generic term, for ‘the next thing that’s going to solve the problems we’re trying to solve.’”

Next the panel moved on to pinpoint whether Web services has yet become common beyond the firewall, or is still mostly being used for intracompany use.

Chappell noted that in his experience there is about an 80:20 divide in terms of adoption. “80% is within the corporation’s control, and 20% involves the public Internet (the Web) – dealing with other business partners, for example.” Brunell agreed that mission-critical apps were still “few and far between,” adding, “That’s why we are all coming to these conferences.”

Microsoft’s White noted that on the contrary he had seen mission-critical things happen inside Web services. “We’ve only just gotten there,” he said, “but I have absolutely seen mission-critical Web services in our customer mass.” Not out in the B2B space, he conceded.

JBoss Group’s CTO, Scott Stark, pinpointed one crucial piece of the jigsaw that’s still missing: “Single Sign-On is a joke, I have about 35 accounts; no one has an agreement yet on a one-stop solution, and no enterprise technology can surmount that. J2EE is still basically a middleware technology,” Stark continued, “it’s not out there bridging enterprises.”

The bridging role, then, remains perfect for Web services. But these things take time, Stark added. “Developers are going to have to get comfortable with Web services first: J2EE has taken 7 years to become a reasonably accepted technology.” He pointed out that XML wasn’t without its shortcomings. “XML is a double-edged sword. My head starts spinning after I’ve read the 10 different XML Schemas. So the usual technology curve also impedes the adoption of Web services. But that’s just the nature of the beast.”

Asked if XML might be replaced, White explained that one of the problems is that good tools are often the last thing to appear after a “technology burst” such as the one we are seeing around Web services. “I’m not a seer,” White said, but the key to widespread adoption of any new technology is completion of the specs (we’re there), demos (we’re getting there), and then the tools (they’re coming).”

JBoss’s Stark agreed. “XML isn’t going anywhere. Before there was IIOP and it went nowhere. Clearly XML is the only technology, however complex it might be, that’s tried to address the problem. Besides, IIOP was even more complicated, and writing, say, a TCP/IP stack, is not a productive endeavor.”

Stark then minted the phrase of the conference. “People have more comfort now with distributed programming; it’s a tool for the times.”

# Industry **News**

### STMicroelectronics/Kudelski Group to Provide 3G Java SIM/USIM Card Solution

(*Geneva / Cheseaux, Switzerland*) – STMicroelectronics and the Kudelski Group have announced the signature of a framework development and license agreement to provide a 3G Secure JavaCard system solution. This system solution consists of a background compatible 3G SIM/USIM card with a design that enables telecom operators as well as third-party application providers to design and deploy new mobile services with the utmost security features.

The jointly developed solution represents an open and standard-compliant card in the telecom market and will be deployed in the first half of 2004.

www.st.com
www.nagra.com

### esmertec Brings to Market MIDP2.0 TRUE BREW-Tested and Java-Certified Solution

(*Geneva*) – esmertec has announced the availability of a market-ready MIDP2.0 TRUE BREW-tested and Java-certified solution. The Java Virtual Machine (JVM) solution enables Java applications to be deployed like any other application that's offered via QUALCOMM's BREW system. The solution successfully passed the TRUE BREW and J2ME CLDC MIDP2.0 compatibility testing, thus the implementation complies with both standard platform requirements and is now available for wireless operators.

www.esmertec.com

### WebMethods to Acquire The Mind Electric

(*Fairfax, VA / Dallas*) – WebMethods, Inc., has announced that it will acquire The Mind Electric (TME) and its flagship GLUE software, a Java-based platform for building Web services.

WebMethods, a maker of software-integration technology, also said that Graham Glass, the founder, chairman, and chief architect of Dallas-based TME, will now serve as WebMethods' chief technology officer.

www.webmethods.com

### Actuate Integrates e.Report Engine with BEA WebLogic Workshop 8.1

(*South San Francisco*) – Actuate Corporation, a provider of scalable business intelligence applications, has announced the Formula One e.Report Engine for WebLogic Workshop, a new product developed by the Reporting-Engines division of Actuate Corporation, an embedded reporting solution for the J2EE platform. The e.Report Engine for WebLogic Workshop offers BEA developers a fully integrated Java reporting toolset that can be used as a part of every project to design, preview, compile, and deploy reports for any application without leaving the BEA WebLogic Workshop environment.

www.reportingengines.com

### Parasoft Releases Jtest 5.0

(*Monrovia, CA*) – Parasoft, a provider of Automated Error Prevention software solutions, has announced the general availability of Jtest 5.0, a development product that automates all aspects of Java unit testing and coding standards compliance. This latest version is equipped with new JUnit test generation capabilities, automated code correction capabilities, and other features designed to simplify team-wide Java error prevention.

www.parasoft.com

### BEA Offers Up New Enterprise Security Architecture

(*San Jose, CA*) – BEA Systems, Inc., has announced that it is addressing a major IT challenge for enterprises with BEA WebLogic Enterprise Security – an application security infrastructure solution. It provides a comprehensive, easy-to-manage, single-security architecture that covers most security services such as authentication, authorization, identity assertion, role and credential mapping, and auditing. The software aims to provide a single point of authentication for enterprise applications. www.bea.com

## Using **Java Generics**

–continued from page 37

```
javac -J-Xbootclasspath/p:JSR14HOME\gjc-
rt.jar -bootclasspath JSR14HOME
\collect.jar;JDK141HOME\jre\lib\rt.jar -
source 1.5 -d classes - classpath classes
FileName.java
```

The Java command contains similarly obscure arguments. The command I used in running the generic examples is:

```
java -Xbootclasspath/p:JSR14HOME\gjc-rt.jar -
cp classes FileName
```

If you're developing on a Unix platform, a make file provided in the JSR 14 download should help you on your way. You can find it in the examples directory in JSR14HOME.

### Conclusion

Changes to the core syntax of Java are rare because of the momentum behind Java. You can't change anything without the risk of affecting millions of lines of code. The generics addition to Java is a welcome one, even with that risk. After years of debating if generics was a good idea and evolving generics compilers, Sun will be sending generics out with Tiger in JDK 1.5. This is a great change because these "generics" or "parameterized types" allow you to create more flexible classes that are still types supported by the compiler.

Generics also solve one of the biggest annoyances in Java code: the constant type-checking with instance-of followed by type-casting. The compiler will catch many of the errors that would have generated ClassCastExceptions at runtime. This will make code more readable and type safer.

The addition of generics is a significant change. Although Java has gone through many versions, none has added as much syntax change as generics, except maybe inner classes. Maybe it's time to move the major version up one and announce JDK 2.0?

### References

- *"Preparing for Generics":* http:// developer.java.sun.com/developer/technicalArticles/releases/generics/
- *JSR 14 Home:* http://jcp.org/en/jsr/detail?id=14
- *Early access release:* http://developer.java.sun.com/developer/earlyAccess/adding_generics/index.html
- *"JDK 1.5 Preview: Addition of Generics Improves Type Checking":* www.devx.com/Java/Article/16024

J2ME
J2SE
J2EE
HOME

# WebLogic
# JRockit 8.1

## by BEA Systems

Reviewed by
**Kirk Pepperdine**

### BEA Systems, Inc.

2315 North First Street
San Jose, CA 95131
**Phone:** 800.817.4232
**Web:** www.bea.com

### Specifications

**Platforms:** Window NT/2000/XP,
Red Hat Linux

### Test Platform

Compaq Armada M700, 850 MHz Intel Pentium III processor, 20GB disk, 576 MB RAM, Windows 2000 service pack 3

The JRockit engineers made two assumptions when they first designed JRockit. First, server VMs run for a long time and, second, memory is cheap and plentiful. This motto still rings true in BEA's offering of the 8.1 (J2SE 1.4.1_03) version of this product. And, unlike the more familiar JVMs, this VM comes with a face.

### Acquiring and Installing JRockit

JRockit runs on the MS Windows and Red Hat Linux platforms and is available in a 25MB download. The install was as uneventful as all installs should be. As expected, the directory structure followed the standard JDK/JRE structure. There were some differences in the content, but all of the standard tools were present. What is missing are the familiar Javadoc and Java source JAR. Not having the Javadoc was not an issue as it's the same one that's provided by the JDK and the quantity of online documentation is overwhelming. In addition to ample information on how to tune the JVM, it included a guide that described how to extract the JRE from the distribution.

### Using JRockit

The two tools that I use on a regular basis are IntelliJ IDEA and Ant. Configuring IntelliJ to use JRockit required only a few mouse clicks. The following code shows how to set the JVM attribute so that Ant will use JRockit. Unfortunately, the Javac task has no such attribute and JRockit is not in the list of available compilers. The easiest way around this limitation is to use the exec task to invoke the compiler.

```
    <target name="listener">
        <java classname="com.jpt.bench-
marks.jrockit.EchoServer"

jvm="${JRockit.home}/bin/java" fork="yes">
            <classpath>
                <pathelement
location="classes"/>
            </classpath>
        </java>
    </target>
```

JRockit supports all the standard VM options and a number of the more "standard" nonstandard –X options (such as heap size and heap max size).

### JRockit Features

The reason for using this VM is its feature set. Curiously enough, one of the most interesting features is that the VM lacks an interpreter. As a result, the VM must JIT every method that it encounters. This does result in longer start-up times and the VM will consume more memory. The upside is that once started, your server will run at speeds that are comparable to a natively compiled executable. HotSpot will still aggressively compile the well-trafficked portion of your application.

Other features of the VM include four garbage collectors, thread local object allocation, thin threads, and a rich set of verbose printing options. The garbage collectors include a generational stop and copy single-spaced concurrent, generational concurrent, parallel, and single-spaced stop the world. It's the latter two GC algorithms that are recommended for production systems. Included in the documentation is a very useful discussion on how to choose between them.

By default, each thread is allocated its own private space in the heap, and objects are in this space. While this feature will result in your application consuming more memory, it eliminates the need to synchronize on a global heap. If memory does become an issue, you can configure the VM to allocate on the global heap.

The verbose printing of information includes memory system, garbage collection, class loading, code generation, CPU, and code optimizations. Normally, I would spend some time on the usefulness of this type of information, but that discussion is cut short in favor of the unique set of management and monitoring capabilities that are included with this VM.

### Managing and Monitoring the JRockit JVM

The current set of management and monitoring APIs in the JRockit JVM are proprietary. Having said this, they are actively involved with the JSR-174 expert group, busily defining a set of specifications for APIs to manage and monitor a JVM. You have to believe that JRockit's still maturing management console is a demonstration of the future of VM technology.

As is the case with the JPDA, the JVM must be started with the monitoring turned on. As can be seen in Table 1, the associated overhead is barely noticeable until the console manager is attached. Once attached, the console did consume more cycles than was expected in this test, though in others, the resource consumption was much lower.

| | |
|---|---|
| No monitoring | 100% |
| Monitoring turned on | 99.5% |
| Console manager attached | 91.4% |

**Table 1** Echo Server running in JRockit (sustained TPS)

The good news is, with this VM we can afford to turn on monitoring in production and attach a console when the need arises. The console displays current readings on CPU, memory, heap, and garbage collection. Figure 1 is a complete summary of heap, memory utilization, and GC statistics. The main panel includes a CPU utilization gauge.

In addition to these statistics, the console supports the notification framework, which listens on aspects and triggers rules whenever the specified set of conditions is met. A rule consists of a trigger and an action. The trigger defines the conditions that must be met before the action will be executed. The action is simply what needs to be done when the trigger fires. The notification framework offers a nice point of extensibility. Developers are free to define their own triggers and actions.

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

*kirk@javaperformancetuning.com*

Attaching a JRockit Runtime Analyzer recorder to the VM allows you to collect runtime information that can be analyzed at a later time. In addition to heap and GC statistics, the recorder captures method optimizations and the execution profile. An example of the detailed GC statistics can be seen in Figure 2.

While the ability to monitor was limited to a few statistics, there was a good level of detail on heap and memory utilization. One capability that I've used in other tools that was missing from this analysis tool was the ability to overlay and manipulate graphs. This feature is very useful when you're attempting to correlate events. As basic as the presentation was, it did offer a lot of data that is otherwise very difficult (if not impossible) to obtain.

## Conclusion

With all the emphasis on performance, it's not surprising that this VM consistently obtains the best results in the SPECjjb2000 benchmarks. It is this drive for performance that has motivated the JRockit engineers to build tools to monitor and manage the VM. And now, these tools are providing us all with insight into the future of how VMs will be monitored and managed. Though the current console is more about monitoring than managing, the future will include such capabilities as being able to alter the choice of GC to altering the size of the nursery.

In creating a server-centric VM, the engineers at JRockit have purposely traded memory for performance, which is a wise choice in this reviewer's humble opinion.

## References

- http://e-docs.bea.com/wljrockit/docs81/
- www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/jrockit/
- www.spec.org/jbb2000/results/res2003q3/

## Snapshot

**Target Audience:** Developers, system administrators, project managers

**Level:** Intermediate

**Pros:**

- Certified J2SE 1.4 compatible
- High-performance, server-oriented VM
- Integrated monitoring system
- Well-documented feature set
- Highly configurable

**Cons:**

- Consumes more memory than a traditional VM
- Slower startup

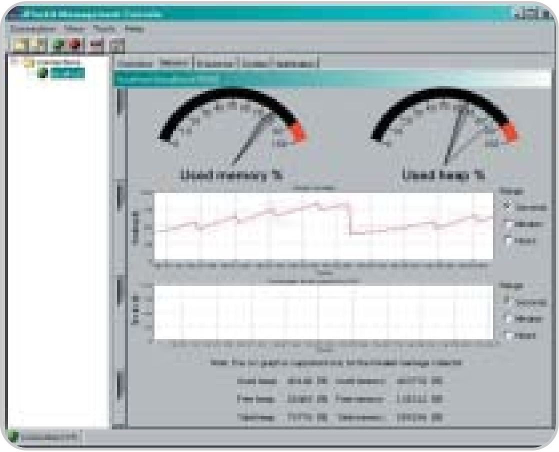


**Figure 1** The memory panel

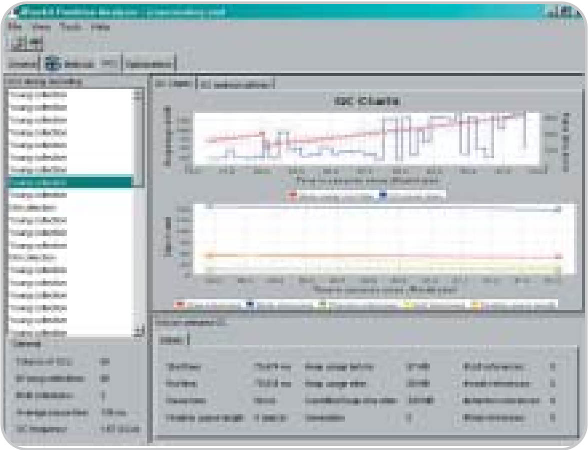


**Figure 2** Heap statistics

# From Within the Java Community Process Program

Onno Kluyt

## From new JSRs to final APIs

Welcome to the November edition of the JCP column! Each month you can read about the Java Community Process: newly submitted JSRs, new draft specs, Java APIs that were finalized, and other news from the JCP. For November I'll be covering a handful of new JSRs, several final JSRs including one rather long-running one that has now reached the finish line, a plug for ApacheCon, and a report on the first phase of this year's EC elections.

### New JSRs

Since the writing of last month's column, four new JSRs were submitted by JCP members. The first new JSR of this year was number 203, and the JSR count is now at 232; thus the community keeps running like clockwork and is on schedule to again hit the average of 40 to 45 new JSRs per year.

This month Siemens has submitted two JSRs. JSR 229, Payment API for J2ME environment, will be developing an API to initiate payment transactions and methods to allow service providers to support different payment instruments. JSR 182 focuses on payment interactions with Web-based services. As such, it can be viewed as a payment instrument implementation for which JSR 229 would provide the framework. JSR 230, Data Sync API, proposes to provide a mechanism for J2ME applications to synchronize data stored on the device with data stored on a server. The JSR aims to develop a high-level API that can plug into a number of underlying synchronization protocols such as SyncML. Also for J2ME technology, Nokia and Motorola have submitted JSR 232, Mobile Operational Management. The proposal is to provide functionality that allows devices based on CLDC and MIDP 2.0 to install and remove components on demand. This gives developers the opportunity to create applications as interoperable and shareable components; it also creates opportunities for providers, manufacturers, and others when these components can be deployed aftermarket and across a wide range of devices.

New in the J2SE environment is Sun's JSR, JSR 231, Java Bindings for OpenGL. The proposal describes the development of Java bindings to the native 3D graphics library, including all core GL calls and the GLU library. This provides the Java developer with access to hardware-accelerated 3D graphics in a portable and open standard way. It will be delivered as an optional package on top of the J2SE platform.

### Final JSRs

There are three final JSRs that I want to cover here. To start off, JSR 97, JavaHelp API, is one that I am quite fond of. While it took some time for this JSR to finish, it delivers functionality that is especially useful to tool vendors and developers of complex desktop applications in general. It's a rich help system aimed at both developers and authors. It's also a nice example of the Java Foundation Classes technology. Also for the J2SE environment, JSR 160, JMX Remote API, has successfully completed. This JSR adds client-side APIs for a so-called Java Manager to discover and access JMX-based agents. This complements JSR 3, which provided the API for management agents and services.

On the J2EE technology side, JSR 124 went final. The J2EE Client Provisioning Specification allows a J2EE server to discover suitable client applications available for delivery, to monitor the delivery of a client application, and to separate the provisioning of various client applications from each other. The API supports several client platforms such as J2ME MIDP and Java Web Start–enabled applications.

### Upcoming Birthday Party

The JCP was launched in December 1998, so in one month it's the community's 5th birthday. To celebrate, the JCP Program Office will be at ApacheCon US 2003 in Las Vegas from November 16–19. The JCP is sponsoring the event and putting together a few activities at the conference to call attention to this milestone. I invite you to come along and discuss with me how the JCP has evolved since December '98 and what directions it can/should take in the next five years.

### The Elections

On October 14 the first phase of the EC elections were completed – the ratification vote by the JCP membership on the nominations by Sun. For the ME EC, Matsushita, Motorola, Siemens, and Vodafone were ratified. For the SE/EE EC, Fujitsu, HP, IBM, and Oracle were ratified. Congratulations to these members and, in addition, a welcome to Vodafone to the ME EC. The voting for the second phase of the elections starts on November 1. These are the self-nominated seats. For each EC there are two seats up for reelection. For ME EC these are BEA (term expired) and Zucotto Wireless (out of business); for SE/EE EC these are Doug Lea (term expired) and Cisco (withdrawing from the EC). There is no limit to the number of terms a JCP member can serve on the EC, so I expect that both BEA and Doug Lea will run for reelection. You can find out more about the elections at PriceWaterhouseCoopers' Web site at http://jcpelection2003.org.

• • •

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

**Onno Kluyt** is the director of the JCP Program Management Office, Sun Microsystems.

*onno@jcp.org*

# Next Month

## Non-Stop EJB Services

Service-oriented architectures (SOA) have numerous benefits: reuse of business logic by many clients, location transparency of business logic, simplified unit testing, better scalability through distributed and load-balanced processing, and composition of new services from existing services. As many new SOA applications are now developed on the J2EE platform, a problem arises: how to maintain 100% availability while deploying maintenance fixes and new versions of the services.

### It's the Aspects

Aspect-Oriented Programming (AOP) is a promising new paradigm that came out of Xerox PARC a few years ago and is just now becoming mature and mainstream. As a natural complement to object-oriented programming, it has the potential to ease the management of complex systems and make their organization much more intuitive, extendable, and flexible.

### Building a Connected MIDlet

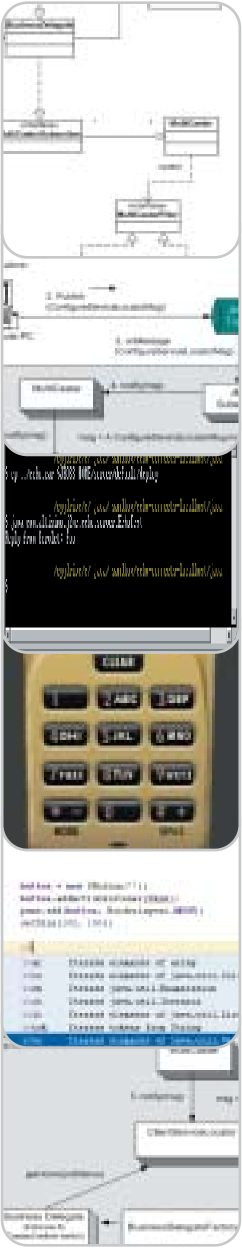
One of the most powerful aspects of J2ME is connected mobility: you're no longer tied to your desk to accomplish many vital tasks. You can carry everything you need in your pocket, send an e-mail while standing in line in the grocery store, or check the latest stock figures while at a baseball game. This article will show you how to build an Internet-enabled mobile application and illustrate the considerations that must be taken into account during design and development.

### IntelliJ IDEA 3.0 by JetBrains, Inc.

The integrated development environment (IDE) makes up a large part of the tools in the toolkit of a modern software developer. Java projects have a complex development process, especially if the project is to be developed using open source and J2EE technologies. A smart and efficient IDE plays an important role in making developers more efficient and productive in doing their tasks and meeting project deadlines. This review will discuss one such smart IDE – IDEA 3.0.

### Teamstudio Analyzer for Java

What is every Java developer's nightmare? Maintaining code, even if he or she has written it. Code is often chaotic and incomprehensible, mostly due to nonuniform coding styles. For decades, premier software vendors realized that uniformity in projects cannot be assured without additional inspections during development. Teamstudio is one of these software vendors. Teamstudio Analyzer for Java automatically inspects your code and provides control over a uniform coding style within your projects.



# Software Development: Science or Art?

Henry Roswell

Many of the problems related to software development are at the individual level, with those who create bad code rather than with any specific technology issue. Therefore the goal of anyone staffing a project is to attract employees most likely to ensure success. The infamous 1968 study by Sackman, Erikson, and Grant, "Exploratory experimental studies comparing online and offline programming performance," concluded that productivity variation between good and bad developers was a factor of 10. The test was based on how quickly their subjects could write a program to solve a maze algorithm, and implicit in this was the assumption that the coders who solved it fastest were superior. It's not an illogical conclusion to make. Most computer science faculties do the same when they grade students based on their ability to write sort routines or their intimate knowledge of Knuth's programming fundamentals.

Frederick P. Brooks in *The Mythical Man-Month* writes, "The hard part of building software is the specification, design, and testing of the conceptual construct, not the labor of representing it." In other words, it's the thought process that is essential to success, rather than coding horsepower. If the act of programming is more aligned to a creative process than to one requiring a traditional engineering discipline, why is this not encouraged and recognized more? The most successful teams I've worked on are usually composed of well-rounded individuals who aren't chasing the latest technology fad to pump up their résumé. These people have both feet firmly on the ground. They stay focused on delivering sensible solutions to the immediate problems.

Writing good software involves foregoing gratuitous complexity, concentrating rather on such concepts as reuse, maintainability, and reliability. Often those most qualified for this aren't professional computer scientists but instead graduates of other disciplines who, instead of focusing on being one-person programming machines, have developed their analytical, creative, and communication skills while at college. It's hard to formalize what traits good programmers need, but I think people interested in subjects outside of computing, the musically minded for instance, make the best coders.

Mathematics disciplines are often separated into pure and applied: number theory versus building a bridge that won't fall down. Perhaps computer science needs the same, so that future generations of programmers will be more grounded in the practicalities of building and maintaining application code. The two most important movements in software of recent years – design patterns and extreme programming – were not born out of any academic research institute but instead came from experienced developers looking for ways to improve the production process.

In "Hackers and Painters", Paul Graham draws parallels between a programmer and an artist. The artist doesn't need to know the chemistry of paint, but instead is responsible for knowing how to mix and match colors to create the finished canvas. The best coders likewise are not immersed in esoteric computing theory, but have a picture of the finished product in their minds and understand the techniques required and the tools available to arrive at completion.

In terms of degree results and employment placements, the safe route for college computer science departments is to recruit mathematically minded students. These can be drilled with little effort, or thought, frankly, into the next generation of speed coders. Each year they collect their degrees only to join business IT departments or software companies. Computer science was born out of a marriage of mathematics and engineering, but surely it's time for the faculty to take a step away from academic roots and try to attract and nurture the more practical side of programming. Universities must attract creative individuals as the next generation of programmers, and dispel the myth that successful software development is a sport practiced by mathmo nerds. Instead it's more akin to poetry for machines, with computer language syntax being the artist's medium.

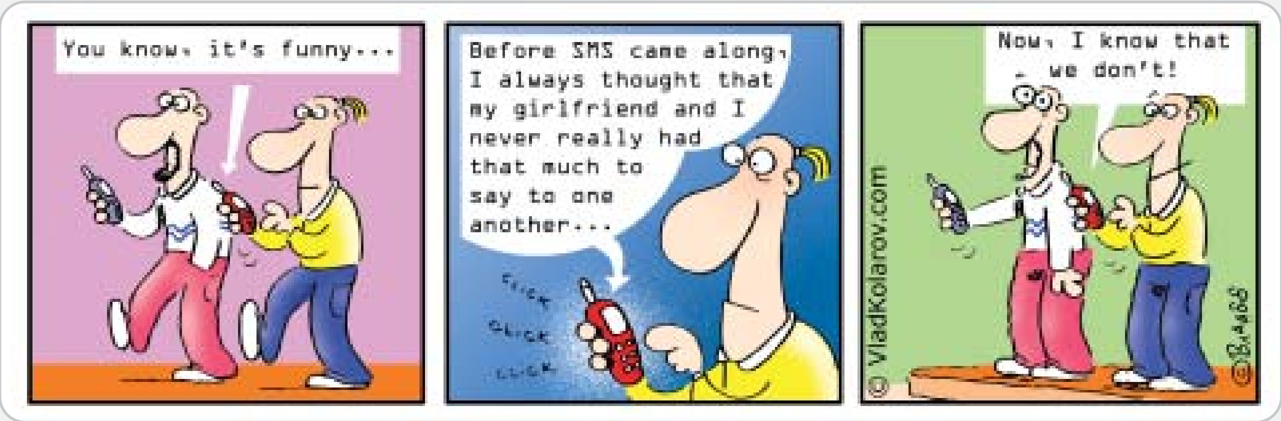
**Henry Roswell** is a veteran consultant who would like to think he's seen it all, but is constantly amazed by new events everyday.

*henry@sys-con.com*

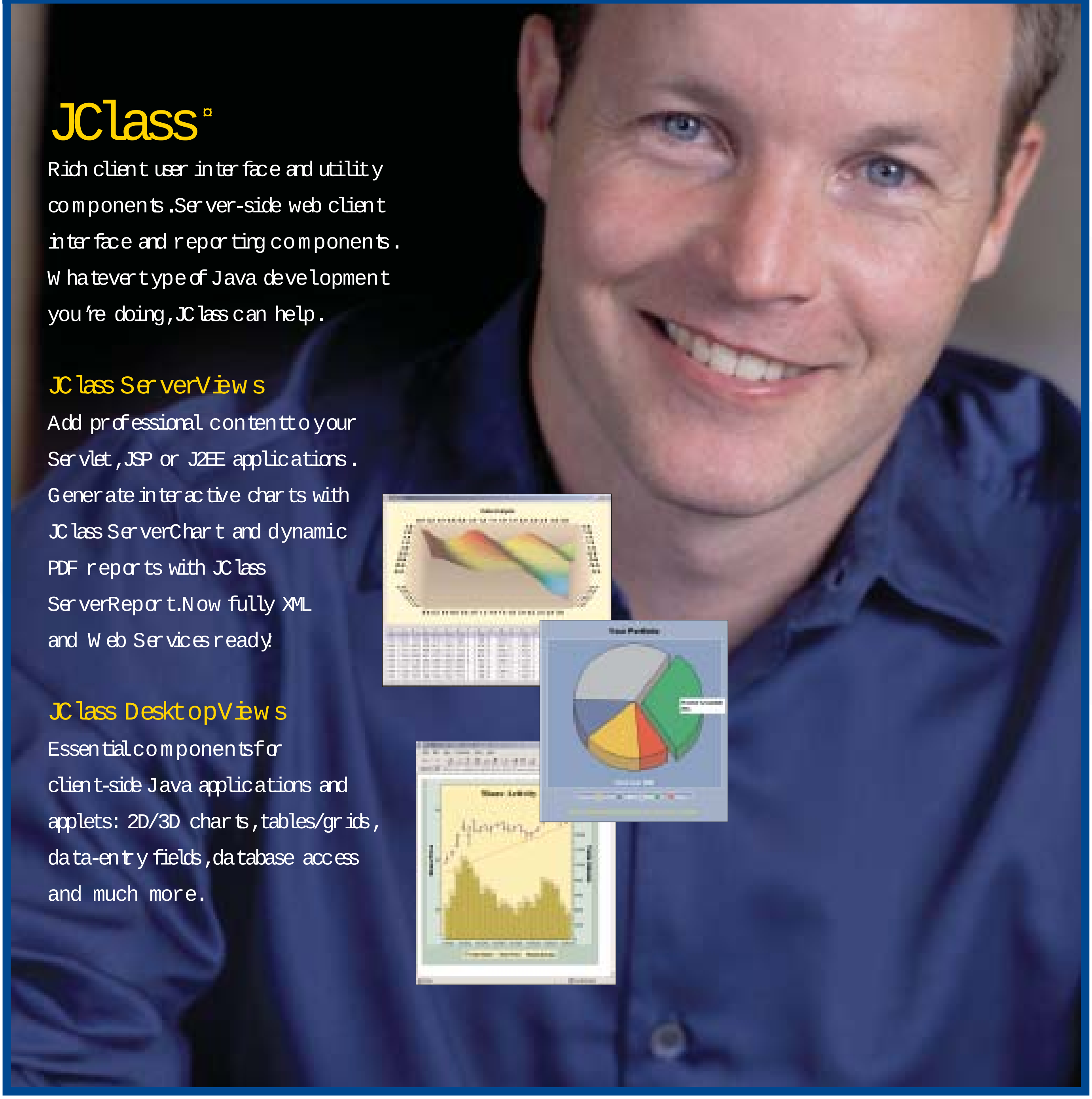
## References

- *Sackman, Erikson, and Grant:* http://portal.acm.org/citation.cfm?id=362858&coll=portal&dl=ACM&CFID=13007643&CFTOKEN=14286366
- *The Mythical Man-Month:* www.aw-bc.com
- *Hackers and Painters:* www.paulgraham.com/hp.html

Java Dudes


You know, it's funny...
Before SMS came along, I always thought that my girlfriend and I never really had that much to say to one another...
CLICK
CLICK
CLICK
Now, I know that we don't!
©VladKolarov.com

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



## JClass¤

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

### JClass ServerViews

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

### JClass DesktopViews

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

QUEST SOFTWARE®

Evaluate and experience JClass today – visit:

http://java.quest.com/jclass/jdj