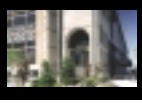


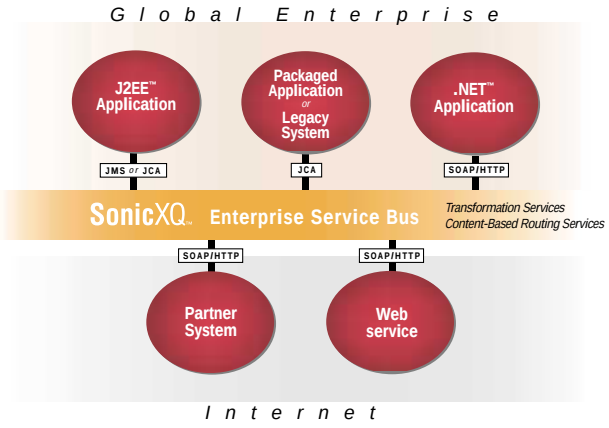
JAVA™ DEVELOPER'S JOURNAL
The World's Leading Java Resource
December 2002 Volume:7 Issue:12

JAVADEVELOPERSJOURNAL.COM

Everybody's focused on exposing applications as Web services while letting someone else figure out how to connect them. We're that someone else.

Introducing SonicXQ™

Integrate with ease, extend at will.™



Most companies are focused on exposing individual applications as Web services, but haven't yet considered how to effectively integrate or manage them. And that's no small task.

SonicXQ is the first product to deliver on the vision of the Enterprise Service Bus: a standards-based, service-oriented architecture that reliably and securely integrates enterprise applications through Web services and JCA technologies. SonicXQ includes configurable XML processing services that allow applications and business partners to easily interact over the SonicXQ bus, eliminating the need for point-to-point integrations and the expensive centralized solutions that implement them.

And because SonicXQ is built upon the proven SonicMQ® messaging backbone, it provides end-to-end reliability and comprehensive security between services. Our patent-pending Dynamic Routing Architecture™ ensures that Web services and distributed processes can scale to meet the demands of today's global enterprises.

Capitalize on the promise of Web services right now with SonicXQ. After all, what good is a Web service if it's an island unto itself?

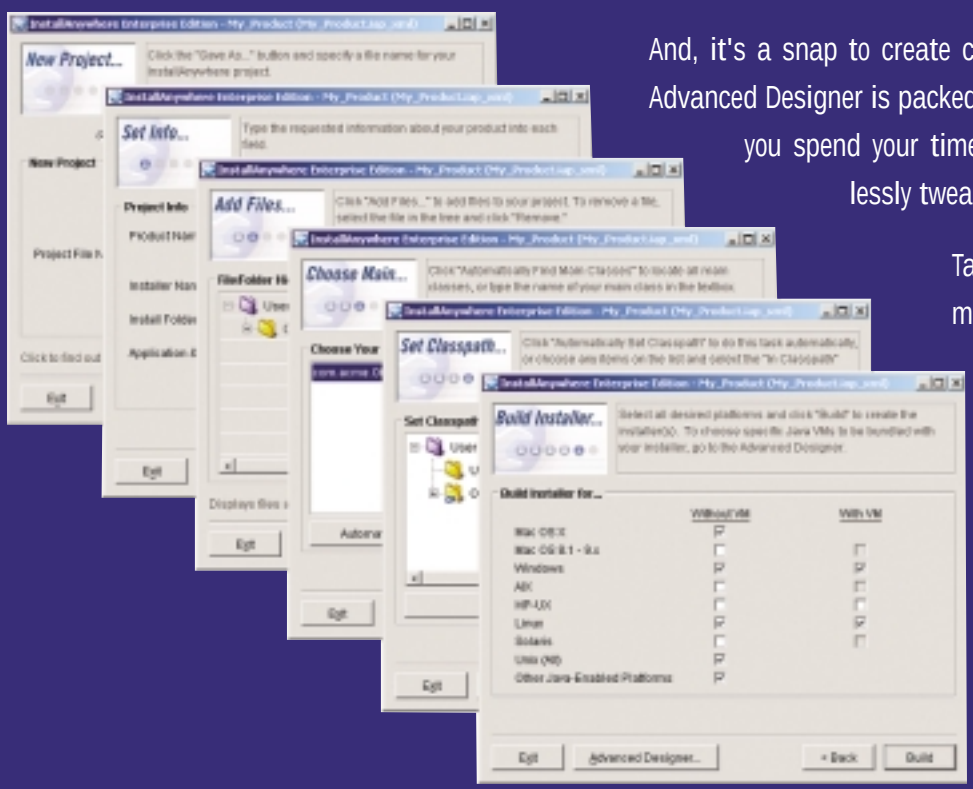
sonic SOFTWARE™

www.SonicSoftware.com/JDJ

their wizards

our wizards

Need an installer in the blink of an eye? With InstallAnywhere, you won't need to take a class at Hogwarts, chant a magic spell, or even drink a potion in order to create your first installer - just click through six panels. InstallAnywhere's Six-Step Wizard creates basic installers in under five minutes, saving you time and money.



And, it's a snap to create custom installers too! InstallAnywhere's Advanced Designer is packed with panels, actions and billboards, so you spend your time creating great applications, not endlessly tweaking your installer project.

Take the "hocus-pocus" out of creating multi-platform installers. Download a FREE fully functional trial version today at www.ZeroG.com and see for yourself why InstallAnywhere is the world's most powerful multi-platform installer.

InstallAnywhere
Zero G
www.ZeroG.com/Install

apple.com/xserve

Porsche has a 420-hp, water-cooled,

Presenting Xserve, one of the fastest 1U servers on Earth. Featuring the exceptional performance of dual 1 GHz PowerPC G4 processors, each with 2MB of Double Data Rate (DDR) L3 cache. Best-in-class storage, with up to 480GB on four hot-plug ATA/100 drives. And best-in-class networking capabilities with standard dual Gigabit Ethernet ports. Xserve also comes with UNIX-based Mac OS X Server software (with an unlimited client license), making it ideal for providing file and print services, mail and web services, streaming digital media and for running database applications. Xserve is also perfect for computational clustering and I/O intensive applications like digital video, high-resolution digital imagery and managing large scientific datasets. And, thanks

The new Apple Xserve.

twin-turbo power plant. We have this.

to a convenient visual feedback system, you can monitor the performance of an entire rack with just a glance. Or, keep an eye on Xserve from virtually anywhere with Apple's next-generation remote management tools. Plus, you can swap parts and replace components at will without tools and get instant help with 4-hour onsite response and 24/7 technical support with AppleCare.* Xserve also provides out-of-the-box support for Mac, Windows, UNIX and Linux clients, three PCI slots (two of which are 64-bit, 66MHz), software RAID mirroring and striping, a VGA graphics card, two USB ports and a host of other features that are far too numerous to list here. Apple Xserve. Take a look under the hood at apple.com/xserve.

apply. For terms and conditions associated with the AppleCare Premium Service and Support Plan, visit www.apple.com/support/products.

MOTOSMARTER

Technology gets smarter every day.
What about you?

Join the **Motorola Developer Program** and access the up-to-the-minute training you need to develop killer apps for Motorola devices and platforms. Code smarter and get applications to market faster, whether you're a beginner or an expert looking to master the latest Motorola products. Online training? We've got it. Onsite training? Yep. Classroom training? Absolutely. J2ME™, Java™, tools? Sure, child's play. To get specifics on class times, platforms, and schedules, visit our site and click on Training.

>> *Join today!* **Visit www.motorola.com/developers/wireless**

For a limited time, MDP is offering free, self-paced, online training for all members.

MOTOROLA
intelligence everywhere™

MG90950A-JDJ0902

JAVA DEVELOPER'S JOURNAL

INTERNATIONAL ADVISORY BOARD

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

EDITORIAL

EDITOR-IN-CHIEF: ALAN WILLIAMSON
EDITORIAL DIRECTOR: JEREMY GEELAN
EXECUTIVE EDITOR: NANCY VALENTINE
J2EE EDITOR: AJIT SAGAR
J2ME EDITOR: JASON R. BRIGGS
J2SE EDITOR: JASON BELL
PRODUCT REVIEW EDITOR: JIM MILBERY
FOUNDING EDITOR: SEAN RHODY

PRODUCTION

PRODUCTION CONSULTANT: JIM MORGAN
ASSOCIATE ART DIRECTOR: LOUIS F. CUFFARI
ASSOCIATE EDITORS: JAMIE MATUSOW
GAIL SCHULTZ
JEAN CASSIDY
ASSISTANT EDITOR: JENNIFER STILLEY
ONLINE EDITOR: LIN GOETZ
TECHNICAL EDITOR: BAHADIR KARUV, PH.D.

WRITERS IN THIS ISSUE

BILL BALOGLU, JASON BELL, KEITH BRAITHWAITE, JASON R. BRIGGS, CARL BURN, KARL MCCABE, PHILIP MILNE, BILLY PALMIERI, BILL RAY, TONY ROBINSON, RICHARD ROSS, BILL ROTH, AJIT SAGAR, PATRICK SMITH, JUSTEN STEPKA, JEREMY WAKEFIELD, CRAIG WALLS, ALAN WILLIAMSON, JOE WINCHESTER

SUBSCRIPTIONS:

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

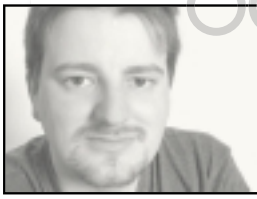
SUBSCRIPTION HOTLINE: SUBSCRIBE@SYS-CON.COM

COVER PRICE: $5.99/ISSUE
DOMESTIC: $49.99/YR. (12 ISSUES)
CANADA/MEXICO: $79.99/YR. OVERSEAS: $99.99/YR.
(U.S. BANKS OR MONEY ORDERS). BACK ISSUES: $10/EA., INTERNATIONAL $15/EA.

EDITORIAL OFFICES:

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

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



ALAN WILLIAMSON EDITOR-IN-CHIEF

Stuff and Nonsense

What an exciting time in the land of Java late October through early November proved to be. During this time The Middleware Company released its now infamous report pitting J2EE against .NET. Why infamous? Well, this particular report suggested that .NET was indeed faster when running Sun's reference PetStore application. But don't despair, as we show in this issue, the benchmark was fundamentally flawed and should never have seen the light of day. I encourage you to read Rickard Öberg's analysis in this issue that shows that The Middleware Company's J2EE implementation of PetStore never stood a fair chance against the Microsoft-optimized PetShop.NET.

Aren't you getting sick of all this propaganda? I know I am. At the end of the day, comparing .NET with J2EE is complete and utter nonsense. It's like comparing SQL to Oracle. Makes no sense. .NET is a full-fledged end-to-end product and J2EE is a framework, an API if you want. It doesn't actually implement or provide anything. It's up to the individual application vendors, such as BEA, IBM, and Oracle, to provide the real implementation of the framework. What on earth are they comparing when they do .NET versus J2EE? Performance-wise, we can't do a single thing.

If you want to do a product-to-product comparison, that's a different story. Let the likes of WebLogic and WebSphere, with all their high-performance components, go head to head with .NET. Don't cripple them by sticking to the purest J2EE specification, thus avoiding what is essentially an academic exercise that mirrors no real-world application. This is the power and flexibility of the J2EE framework; it enables independent vendors to innovate and extend to ensure the underlying server is optimized as much as possible. For example, features such as connection pooling, caching, load balancing, and native socket polling are not covered by J2EE. Turning off all these features is madness. You wouldn't turn them off in a production environment! Then there is, of course, all the value-adds that vendors put into their J2EE offerings, such as integration, process management, and Web services; all things not truly covered in a benchmark of this ilk.

Microsoft is preying on people's ignorance to gain a hand in a poker game that, when you boil it down, is a one-player game; J2EE has no place at the table.

Believe me this won't be the end of it. We're going to see a lot more of this PR warfare as Microsoft turns up the heat to convince the industry it's a serious player in the enterprise space. The people you really have to watch out for aren't, ironically, people with @microsoft.com e-mails, but their legion of happy disciples who will faithfully hang on Microsoft's every word. But have none of it I tell you. Rise above it. Sadly, The Middleware Company made a major error in judgment by aligning themselves with Microsoft to produce this report.

• • •

Having recently attended OracleWorld in San Francisco, I am happy to report that I have seen the future. And would you believe it, for the second time it was at the Nokia booth.

The sheer genius of Nokia can't really be appreciated via any printed or online medium; you have to touch, hold, and feel the sheer beauty the clever Fins produce. Their next model, 6800, out in January 2003, is all I would ever want in a mobile phone – small but flips around for a QWERTY keyboard (with the screen whipping around with it!), color, and a great-processing platform for Java that will allow us to start developing some meaty J2ME applications. The sheer genius in providing a thumbable, flippable QWERTY keyboard is going to rival BlackBerry devices and PDAs. Check it out at www.nokia.com/nokia/0,5184,4486,00.html.

alan@sys-con.com

AUTHOR BIO

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

J2EE, Democracy, and Sausage

WRITTEN BY
BILL ROTH

Back in the beginning of October, I was dragged into the middle of a raging e-mail argument. The argument was whether J2EE was a success, and if it was too complicated. This was like waving a red cape in front of a Spanish bull. I felt then, as I feel now, compelled to respond.

Full disclosure: I am writing this as a member of ***JDJ***'s editorial board. However, I am an unabashed J2EE partisan, and I remain a Sun employee. While I no longer work on the Java platform, I am proud of the work my team did, and the work the current team is doing now, in defining and evangelizing the Java platform. I still believe that J2EE is the last best hope for freedom from tyranny in the enterprise software arena.

There are two key ideas underlying the Java platform in its entirety. First is the unwavering belief in "Write Once, Run Anywhere." This motivates everything we do on the Java platform. Sometimes, we do things that the community appreciates, such as making the specifications freely available on the Web. Sometimes, it motivates us to do necessary things that are less appreciated, such as creating a licensing architecture to ensure that if an application is written to the standard, it will run anywhere.

The second key idea behind the Java platform is fairness, both to software vendors and the community. This is one of the motivating factors behind the JCP. The JCP is a framework that attempts to ensure that all parties have a fair share in the decision-making process, and that the development of the technology meets the community's needs.

The process is imperfect, but rapidly evolving to address those imperfections. Like the old saying about democracy, it is a rotten system, but it's better than every other system that has been tried in the past. There are those who carp about the process, but they're usually the ones who don't succeed in their particular agenda. The process seeks to meet the needs of the community as a whole.

Two main criticisms are leveled at J2EE. First, it's too complicated. After looking at the specifications, this is a reasonable conclusion. Second, it's too hard to use. From a review of many sample applications, this is also a reasonable conclusion. However, both of these criticisms are misguided and specious.

From looking at the specifications, programmers can reasonably come to the conclusion that J2EE is too complicated. The main problem is that the specifications were not written for application developers. They were written for (and by) the systems software architects at the roughly 80 companies in the world that build enterprise-class transactional software.

It's important to understand that the specifications are written as part of the deliberative process required by the JCP. The specifications that result can be likened to legislation and sausage: the final products look and taste good, respectively, but you don't want to know what went into making them.

As a result of this process, the industry has adopted J2EE rapidly because it provides scalability, it helps integrate diverse applications, and it provides a consistent development API.

The second criticism has emerged as J2EE has expanded its developer base from the small group of highly skilled enterprise developers to a larger body of developers: J2EE is too hard to use.

The basic API of J2EE is what it is. It was our belief that tools vendors would build products that would ultimately make programming easier. The production of tools to simplify J2EE programming has been slower than we anticipated, but recent announcements from all the major systems vendors are encouraging.

In summary, let me be clear. J2EE is something unique in the short history of the computer industry. It's a technology developed cooperatively with an entire industry. It is one of the few times when a group got together, agreed on a direction, and built a billion-dollar industry in a few short years. The progress we've made is substantial, but we have much more work to do.

br@billroth.net

AUTHOR BIO

Bill Roth is group product marketing manager for Sun's x86-based servers. Prior to this, he was group marketing manager for J2EE. Bill has 15 years of industry experience in engineering, sales, and marketing roles. He has an MS in computer science from the University of Wisconsin.

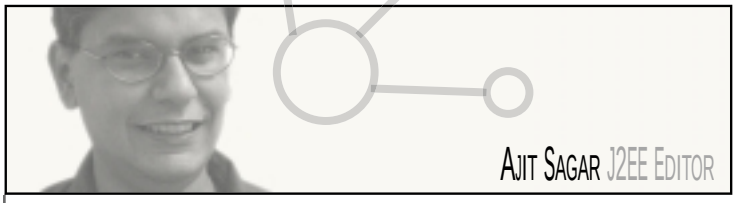
Developing Web Services before WebLogic Workshop.

Developing Web Services after WebLogic Workshop.

If getting dinner via a spear strikes you as less than efficient, you'll appreciate BEA WebLogic Workshop™. This unique development framework eliminates tedious and time-consuming steps, while you build Web services, create new applications and tackle the toughest integration challenges. Plus, you'll have all the support of our dev2dev developer community. To see how BEA WebLogic Workshop can radically simplify your Web services development, download it today at **dev2dev.bea.com/useworkshop.**

bea®

The fastest route to enterprise Web services.



AJIT SAGAR J2EE EDITOR

'Tis the Season for Amalgamations

By the time you get this issue, Christmas will be around the corner. From the J2EE arena, what is on your wish list for the coming year? More sophisticated tools? All-encompassing solutions for your business? More J2EE-related jobs next year? A utopia where J2EE and .NET can live together happily ever after? Better, cheaper, faster environments to build your solutions?

For me, one of the most significant developments this year has been the tighter relationships that open source products have developed with commercial J2EE or related technology vendors, and vice versa. At ***JDJ***, we've been covering a lot of activity in the world of open source vis-à-vis Java. Given the state of today's economy, one of the trends you must have noticed is the strategy that commercial vendors have adopted – embrace open source, embed it in your offerings, and promote a unified, standardized Java environment. This gets the buy-in from the Java community. But more important, Java developers get a better-supported product suite to build enterprise applications.

There is a symbiosis between the different environments right now. As always, established vendors are magnifying their footprints by integrating with products and frameworks that are not a part of their core technology. However, there's a greater focus on adopting open source development to achieve this. This happens at several levels. Let's consider IDEs first.

Most of the leading IDEs, including Sun ONE Studio, WebSphere Studio, JBuilder, IDEA – almost every application server worth its salt – provides varying levels of support for the open source build tool (Ant) and testing framework (JUnit). Others provide support for tools that are not yet de facto standards, like Cactus. At the same time, the J2EE application servers have adopted these tools as de facto standards for configuration and deployment.

To take the acceptance of open source software one level higher, let's look at J2EE IDEs. WebSphere Studio is an example of an IDE that has not only adopted these standards, but is building the next version of their product on an open source framework – Eclipse. Struts is becoming a default standard for most IDEs. This doesn't mean your IDE can just integrate with Struts applications; it lets you build the presentation tier on Struts components that are natively supported from within the development environment.

Now let's look at J2EE and open source from the perspective of business solution and enterprise integration providers. Vendors in this area, as expected, are expanding into each other's space. In the past few years, this expansion was achieved by acquisitions and mergers. Naturally, vendors are a lot more conservative in today's economy. Besides the cost of acquiring a company, one of the concerns is the long-term viability of the company being acquired. Open source again emerges as cost effective and the more stable choice.

webMethods, a leading vendor in the enterprise integration space, recently made a very smart move by planning to integrate the JBoss J2EE application server directly within the webMethods integration platform. J2EE application server vendors have tried to grow into the enterprise integration space via J2EE Connector Architecture or direct integration via native connectors. This recent move by webMethods is an example of integration servers growing into the J2EE application server space.

In the near future, a business solution provider can look at an integrated platform that offers connectivity to legacy systems via Web services, as well as J2EE components that are natively hosted by a single vendor.

Earlier I mentioned symbiosis. On the other side of the coin, the J2EE open source community gains a lot too. The industry as a whole is always a little wary of something that is free. When open source is backed by popular commercial vendors, and much-needed marketing clout is put behind such initiatives, the community has a far greater chance of survival and future growth. The result is a shrink-wrapped offering that should make it onto all our Christmas lists. On that note, I hope all of you have a happy holiday!

ajit@sys-con.com

AUTHOR BIO

Ajit Sagar is the J2EE editor of JDJ and the founding editor of XML-Journal. He is the director of engineering at Controlling Factor, a leading B2B software solutions firm based in Dallas, and is well versed in Java, Web, and XML technologies.

J2EE INDEX

DEVELOPERWORKS

WEB SERVICES PLAY

1] DON'T JUST COMPETE. WIN: Whether you're looking for time-saving tips or simply to download code, there's one source for everything you need to develop Web services apps. It's developerWorks™ and it's the perfect gathering place for developers. That's because it's packed with what you need to simplify building interoperable apps, like tutorials, free code and the latest on technologies and standards (such as UDDI, WSDL and SOAP). Even more, it's the one place you can always go to learn what's fact and what's just hype.

2] WELCOME TO TRAINING CAMP: It's also where you can get the IBM WebSphere Software Developer Kit for Web Services. This Kit has everything you need to get started designing, building and testing Web services apps. To register, visit **ibm.com**/developerWorks/webservices/start

e business is the game. Play to win.™

IBM

Review of 'The PetStore Revisited ...

...J2EE vs .NET Application Server Performance Benchmark'

WRITTEN BY
RICKARD ÖBERG

The Middleware Company (TMC) recently published a benchmark report that compares the Sun J2EE PetStore with a functionally equivalent version created using .NET technologies. The J2EE PetStore version was improved from the original code by TMC employees, while the .NET version was written and optimized by Microsoft employees.

As this summary will show, the report is seriously flawed. One of the big points of the report was that the initial benchmarks and comparisons were done with a version of the J2EE PetStore that was "not properly optimized for performance and not meant to be benchmarked." This report supposedly fixes this since the new implementation has been "extensively optimized for performance and scalability," even claiming that they have rebuilt the PetStore "fully optimizing it for performance."

The Entity Beans

Let's start with the obvious: the use of BMP entity beans. BMP entity beans are a part of EJB's legacy, and were particularly useful in the beginning since there were no advanced CMP implementations around, and semantics and features varied greatly between implementations. In particular, BMP entity beans have something known as the 1+*n* loading problem, whereby doing a finder call results in one SQL call for the actual query, then one SQL call for each entity bean that was found. CMP entity beans don't have this problem since read-ahead can be used in order to eagerly load the beans.

The choice to use BMP entity beans is a factor that on its own seriously impacts the performance of the application. The claim was that the J2EE PetStore now was "fully optimized for performance," but this clearly shows that it was not.

Garbage Collection

Furthermore, the report authors note in many places that garbage collection is an important source of slowdowns in J2EE, and that concurrent garbage collection should be used if possible. While I agree that garbage collection is an important thing to consider, it should be noted that concurrent garbage collection is only preferred if it's important that the end user does not perceive any slowdowns. As is noted in Sun's own VM documentation, concurrent garbage collection yields a lower throughput than regular garbage collection, so in a benchmark you would assume that concurrent garbage collection should not be turned on.

One result of the conclusion that avoiding GC is a good thing is, of course, that you should try to minimize the number of objects created as much as possible. However, the J2EE PetStore is seriously misdesigned in this regard in one particular place: the use of value objects. If an entity bean has not changed, then two calls to the value object method (e.g., getProduct() in ProductEJB) should result in the creation of one value object that's then cached for subsequent calls. This is particularly true when read-only state is used (as in this case), and this is also, after all, how the .NET code works.

Caching

While the J2EE PetStore code goes through some serious hoops to store parts of a category product list in the user session, the .NET code simply caches the entire category product listing in an application-level cache. Read the previous sentence again. I couldn't believe my eyes when I saw this difference between the two implementations. While the J2EE code is doing finders, getting value objects, putting stuff into users' sessions, the .NET code just fetches the objects from the Cache object (that is, as long as it doesn't need to ask the business/data object Product for a category-specific listing during the cache construction). This makes the comparison between the two extremely unfair. This, along with the above mentioned 1+*n* problem, could easily explain the majority of the differences between the J2EE PetStore and the .NET PetStore.

Transactional Settings

We read "in the revised Middle Java PetStore 2.0, all transaction boundaries were revised to minimize the number of transactions." Presumably, this should mean that there's a minimal amount of transactions being started during the execution of the J2EE PetStore. But is this the case?

In the XML descriptor for CatalogEJB (shoppingcart_ejb.xml) we find that all methods run as TX_REQUIRED, i.e., transactions are started for each call. But is this necessary? Some of them deal with returning sets of products and categories. Those items are read-only, so dealing with them should not involve transactions. But, in the "fully optimized" J2EE PetStore, where "all transaction boundaries were revised to minimize the number of transactions," it seems this does not apply.

SQL

One serious issue that dramatically added to the Line Count metric was the absurd absence of PreparedStatement. All SQL queries were built by hand, string concatenating.

Conclusion

As you can see, the J2EE implementation never stood a chance against Microsoft's .NET implementation. There were clearly fundamental 101 errors that make the report completely null and void. You can read my full analysis at http://dreambean.com/petstore.html.

AUTHOR BIO

Rickard Öberg is a recognized J2EE/EJB expert, and contributed to the EJB container implementation of JBoss. He is also a member of the popular open source tools XDoclet and WebWork, and is now working with portal/CMS solutions using AOP technology.

rickard@dreambean.com

analyze the problem

understand the customer needs

define the requirements

control the changes

deliver the right solution

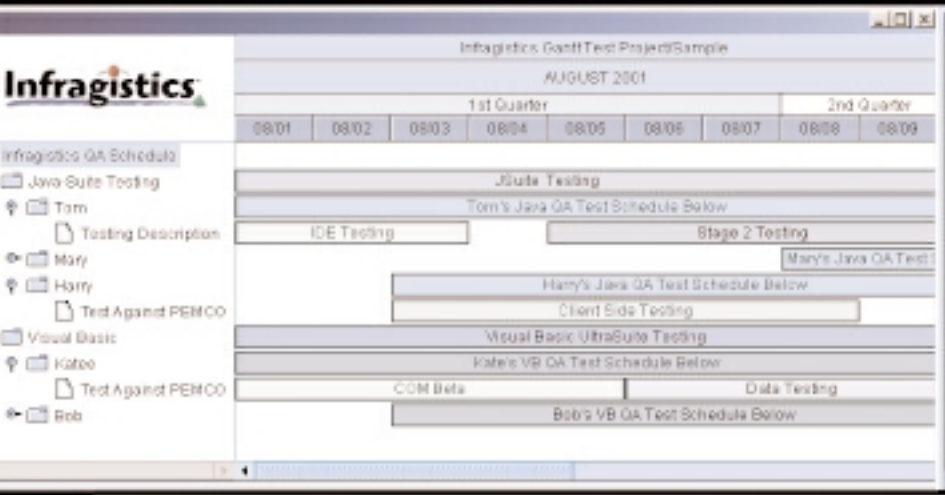
start club for overachievers

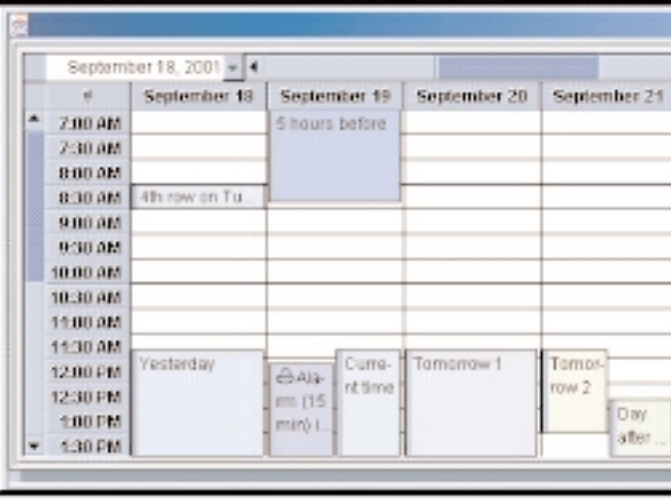
be liberated

A job? A job is something normal people have. You, however, have 20 jobs. You have customers to satisfy. Requirements to meet. Changes to control. And more. But if you're lucky, you also have Rational.® With solutions to help you visually describe and manage requirements. Solutions that automate traceability across your project's lifecycle, so you can control and communicate the impact of changes. And solutions that are flexible, easy to learn in a day and easy to expand over time. Find out how Rational can liberate you and help you satisfy your customers. Go to www.rational.com/offer/roadmap4 today to get your free roadmap to business modeling and requirements management.

Rational®
the software development company

JSuite™ 6.1–Total Java Solution





JSuite 6.1–AWT, JavaBeans & JFC

Applet Wrappers
- Our applet wrappers allow you to manipulate the appearance and/or data associated with each component without any Java code at all, using JavaScript or VBScript.

Infragistics Data Models™
- Advanced n-tier architecture seamlessly connects streaming data over sockets to front-end applications.

Calendaring
- Our scheduling and time management components include a Calendar, DayView, WeekView and DateEdit with drop-down calendar.

Explorer
- Drill-down through hierarchical data structures, with this familiar UI, to display corresponding tables or custom Java panels in the right-hand pane.

Table
- Bind to any data source using flexible Infragistics Data Models. The DataTable is available in AWT and JFC versions.
- The JFC DataTable extends JTable with cell editing, merging, printing and much more.

Editing Components
- A robust collection of components, available in AWT and JFC versions, essential for data input.

Gantt
- Display hierarchical data in the familiar style of a Gantt chart (tree nodes on the left associated with bar chart data display on the right).

Utility and UI Components
- Tab, color picker, calculator, scrollbar and shapes, as well as flexible, customizable image, round and spin buttons.

TreeView
- Create eye-appealing, feature-rich treeviews for web site navigation and Java applications. Available in both AWT and JFC versions.

Chart
- Advanced 2D and 3D rendering engine to quickly deliver stunning charts to any application.
- **We've Added** server side charting (JFC and JavaBeans™) Quickly connect to data from any source with minimal code. Achieve thin client delivery with the look and feel of a rich client application.

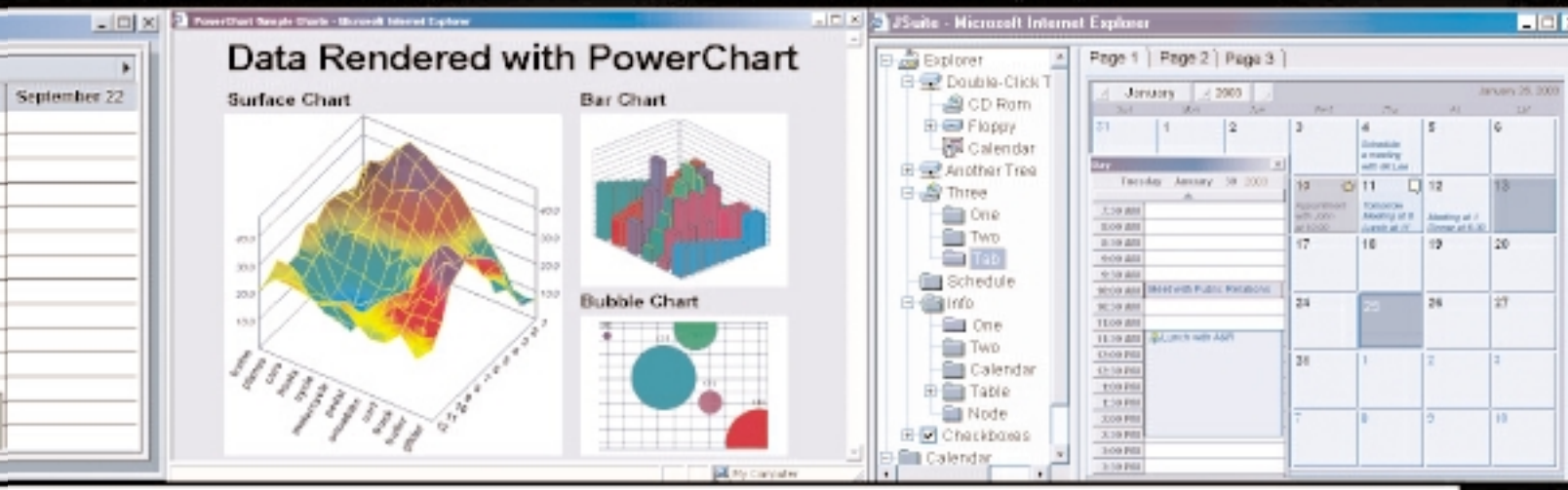
Server Side Classes
- **Access Control List** — Regulate permissions and levels of access based on roles assigned to your application/system user's logon.
- **Password Validation** — Authenticate user ID and password against a directory server or database, with encryption.
- **Connection Pooling** — Maintain a specified pool of connections at startup, eliminating the need for time-dependent "retry connection" code.

Download a free, full-featured trial version!
Order online, or contact us!

www.infragistics.com
800-231-8588

Java COM .NET **infragistics.com/jsuite**

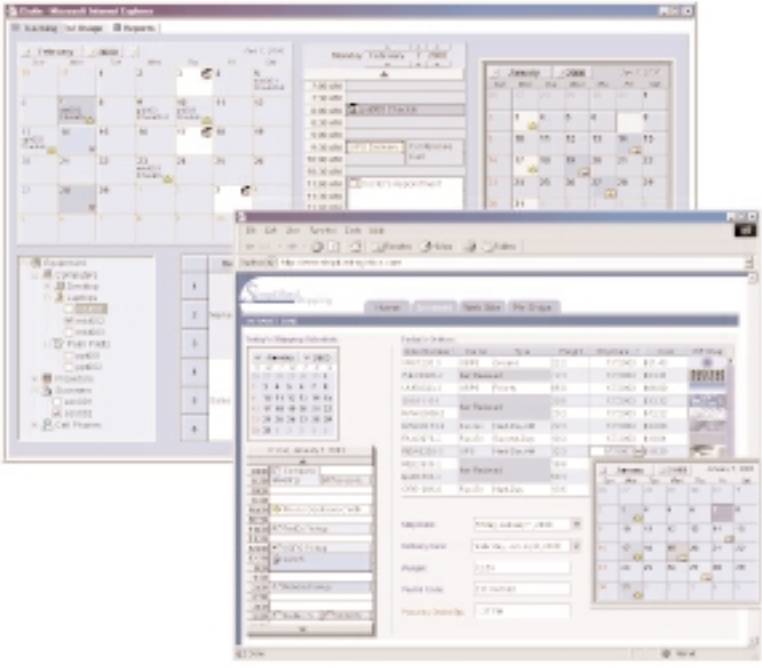
Packaged To Please.



The most comprehensive set of AWT, JavaBeans™ & JFC components

JSuite 6.1

- JSuite provides the most comprehensive set of Java-based components available today. Now we've combined our Java suites to deliver it all: AWT, JavaBeans™ and JFC.
- Create powerful, eye-pleasing applications or browser-based applets with the rich look and feel your users expect.
- We've included everything you need: grid, tree, client-side and server side charting, data explorer, Gantt, calendaring, rich, graphical editing and UI components and convenient utility components.



Source Code for all components included in JSuite — only with subscription or Enterprise Edition!

JSuite 6.1
(includes server side charting) **$795**

with annual subscription
a year's worth of updates, upgrades and new products
→ **includes source code** **$995**

Enterprise Edition
product, subscription as well as guaranteed priority support
→ **includes source code** **$1,495**

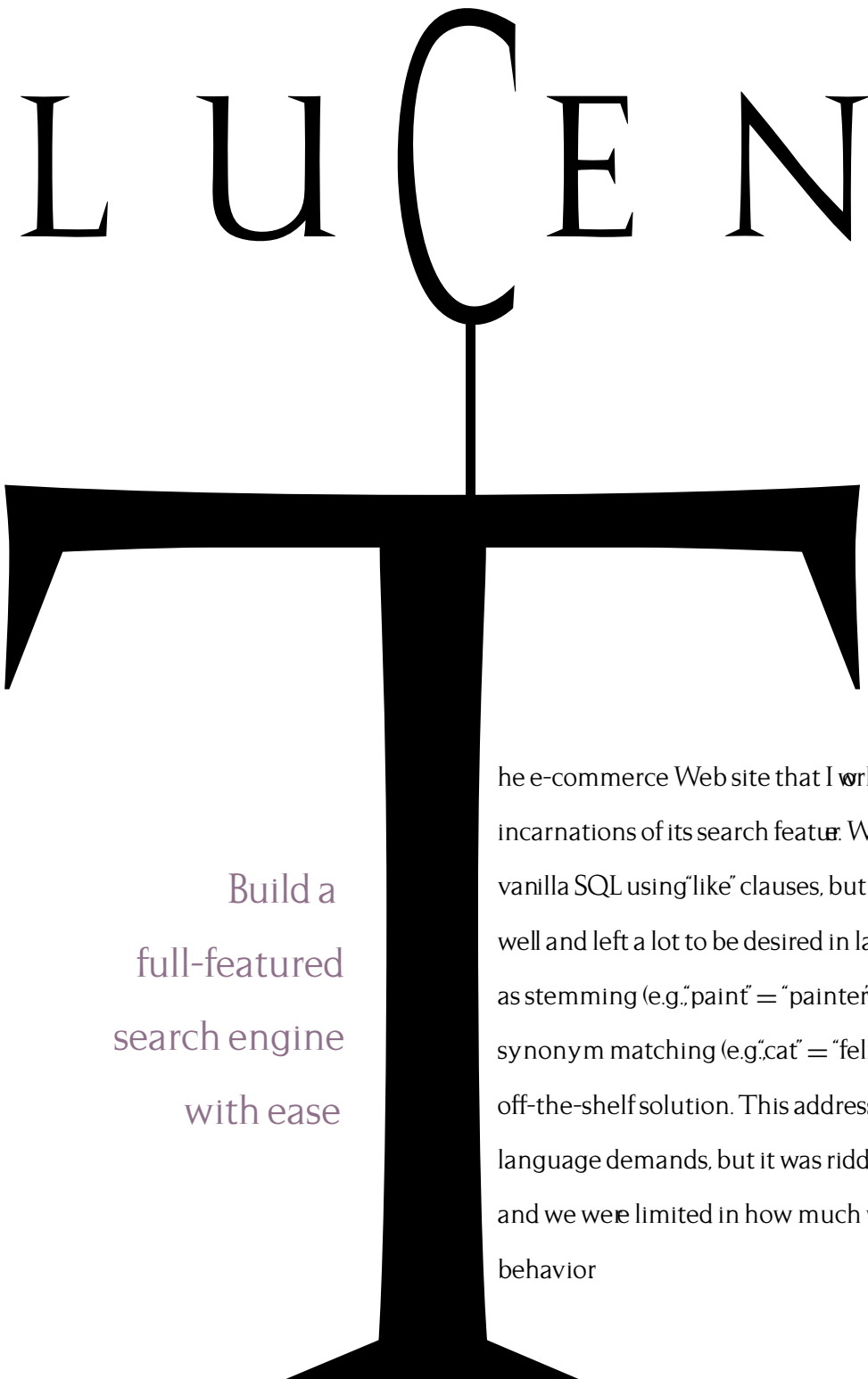
Infragistics™

— **Download a trial version today!**

written by craig walls

search-enable your application with

LUCENE



Build a full-featured search engine with ease

The e-commerce Web site that I work on has seen several incarnations of its search feature. We started with plain vanilla SQL using "like" clauses, but this didn't perform well and left a lot to be desired in language features such as stemming (e.g., "paint" = "painter" = "painting") and synonym matching (e.g., "cat" = "feline"). Next we tried an off-the-shelf solution. This addressed our efficiency and language demands, but it was ridden with strange quirks and we were limited in how much we could customize its behavior

Then we discovered Lucene. Lucene is an open-source search framework from Apache's Jakarta project. As a framework, Lucene provides you with the building blocks you need to build a search engine that meets your specific searching requirements. Lucene is flexible, fully customizable, and amazingly fast.

In this article I show you how to use Lucene to build a search solution for your application. Although my examples will be geared toward an e-commerce application, Lucene is flexible enough to be used on any application whether it's Web, desktop, or CD-ROM based.

I used version 1.2 of Lucene to develop the examples in this article. It can be downloaded from http://jakarta.apache.org/lucene. Lucene is self-contained, so you'll need only a JVM (v1.1.8 or higher) to use it. Place lucene-1.2.jar into your classpath and you're ready to start.

Indexing Documents

To build a Lucene index, first you'll need an instance of IndexWriter. The following lines of code create an IndexWriter for an index located at c:\myindex.

```
Analyzer analyzer = new StopAnalyzer();
writer = new IndexWriter("c:/myindex", analyzer, true);
```

The first argument to the constructor is the path where the index will be written. If the path doesn't already exist, Lucene will create it for you. The second argument is the Analyzer you want IndexWriter to use when tokenizing text. Here I used StopAnalyzer to remove stop words ("and," "or," "the," etc.) from the token stream. The last argument tells IndexWriter whether to create a new index or to add documents to an existing one. Passing true to the constructor will create the index from scratch; passing false will append to an existing index.

Now that you have an IndexWriter, you're ready to start adding documents to the index. The following code creates a simple document that represents a Web page and uses IndexWriter to add it to the index.

```
String url = "http://jakarta.apache.org/lucene";
String content = indexer.retrieveWebPageContent(url);
String keywords = indexer.extractKeywords(content);

Document doc = new Document();
doc.add(Field.UnIndexed("url", url));
doc.add(Field.UnStored("keywords", keywords));
doc.add(Field.Text("content", content));
writer.addDocument(doc);
```

In this example, the document contains the URL metadata for Lucene's homepage, a keywords field that contains search terms to match against in a search, and a "content" field that contains the full content of the Web page.

Once all documents have been added, all that remains is to close the index.

```
writer.close();
```

Although this example adds only a single (hard-coded) document to an index, it serves well as a "Hello World" example of how to create indexes using Lucene. The complete source code for this example is in Listing 1. (Listings 1–10 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

For a more interesting example, suppose you're indexing a product catalog to be searched on an e-commerce Web site. A product is made up of a SKU, a name, a price, and some key-

words to be searched on (see Listing 2). ProductIndexer (see Listing 3) is a convenience class used to add products to a Lucene index.

The constructor for ProductIndexer takes a string that's the path where the Lucene index will be built and a Boolean parameter that specifies whether a new index will be created or an existing index appended. ProductIndexer uses StopAnalyzer for tokenizing text.

The addProduct() method creates an instance of Document and translates the attributes of the Product into document fields. As in the simple example earlier, the "keywords" field is created as unstored so it can be searched upon but is unavailable for retrieval. The other fields are created as unindexed because these fields will be retrieved only after a successful search, not searched upon themselves.

The close() method closes the IndexWriter, making it available for searching. Before closing, however, a call is made to the IndexWriter's optimize() method to have Lucene optimize the index. Although it's entirely optional, it's generally a good idea to call optimize() if the indexing is finished for the time being and no further documents will be added to the index for a while.

ProductDBIndexer (see Listing 4) reads products from a "catalog" table in a relational database (see Table 1 for the products that I used) and uses ProductIndexer to add the products to Lucene's index. ProductDBIndexer takes two command-line arguments: the path in which to build the index and an optional "create" flag to indicate that the index should be built from scratch.

Lucene Index Structure

Lucene indexes are file based. If you look in the directory where you created the index, you'll find several files that define the Lucene index. Depending on how large your index is, you'll see several groups of files where each file in a group has the same name but a different extension. Each of these groups is known as a "segment." Although this article won't delve into the details of how Lucene segments work, it may be interesting to note that IndexWriter's optimize() method optimizes Lucene's index by consolidating all segments into a single segment for more efficient searching.

While IndexWriter is writing indexes, a file called "write.lock" is created. This file prevents other instances of IndexWriter from writing to the index concurrently. Calling IndexWriter's close() method removes this file and makes the index available for writing by another IndexWriter.

Lucene keeps track of each segment in the index using a file called "segments". During indexing, it occasionally becomes necessary for Lucene to update the segments file to keep it synchronized with the segments in the index. While this synchronization is going on, Lucene creates a "commit.lock" file to prevent concurrent updates of the segments file. Once the segments file is in sync, the commit.lock file is removed.

What would happen if you were to write to an index while it's being searched on? You may write to the index (either by adding new documents or re-creating the index from scratch) while it's being searched, but doing so may have undesirable effects on the search results. The worst side effect that I've seen is a document appearing out of order in the Hits collection. Depending on how important the ordering is to you, it may be best to create your indexes off-line (i.e., in another directory) and then rename the directory to become the current index.

Searching

Now that you've built an index, it's time to perform search queries against it. ProductSearcher (see Listing 5) shows how to do this.

INDEX COMPONENTS

A Lucene index is a collection of documents organized in a way that allows quick retrieval of information when arbitrarily queried upon.

Each document (implemented by org.apache.lucene.document.Document) in a Lucene index is made up of one or more fields that are name-value pairs, much like entries in a HashMap. A document can contain as much or as little information as is required to be searched upon. For example, a Lucene document could contain the complete contents of a Web page, text file, e-mail, etc. On the other hand, a Lucene document may contain only a minimal set of metadata, such as keywords, along with a URL, a product SKU, or some other identifying information used to reference a full information source stored outside of Lucene (such as in a file system or a relational database).

Each field in a document can be defined as being any combination of stored, indexed, and tokenized. If a field is stored, its contents are fully retrievable upon a successful search. If a field is indexed, its content may be referenced in a query and searched upon. If a field is tokenized, its content is broken into one or more tokens (or words) prior to being indexed.

Fields can be created using org.apache.lucene.document.Field. The Field class has several static factory methods that make short work of creating field entries. Table 3 illustrates these static methods and the types of Fields that they create.

Why would you want to index a field, but not store it? Consider a field that contains keywords for your document: chances are you'll never display or perform any processing of this field, but you still want to be able to search upon it. By indexing it you're making the field searchable, but by not storing it, you're saving space because the text is not written verbatim to the index. On the other hand, you may want to store some data so that it can be retrieved later but not actually be able to search upon it. In that case, you'd choose a field that's stored but not indexed. When defining your fields, be mindful of what those fields will be used for, and for efficiency's sake choose an appropriate field definition.

To search a Lucene index you need an instance of org.apache.lucene.search.Searcher. Two subclasses of Searcher come with Lucene. IndexSearcher is for searching a single Lucene index while MultiSearcher is used to search multiple indexes at once. Only the product catalog index will be searched, so IndexSearcher is the best choice for this example. It's constructed given the path to the index.

```
Searcher searcher = new IndexSearcher(indexPath);
```

Next you must construct a Query object. The best way to do this is to use the parse() method of org.apache.lucene.queryParser.QueryParser. Create an instance of QueryParser, passing the name of the default field (the field that's searched upon by default) and an analyzer to the constructor. Then call parse() on the QueryParser instance pass-

Static Field Method	Stored?	Indexed?	Tokenized?
Keyword()	Y	Y	N
Text() – Reader version	N	Y	Y
Text() – String version	Y	Y	Y
UnIndexed	Y	N	N
UnStored	N	Y	Y

TABLE 1 Catalog database

ing the query string. An instance of org.apache.lucene.search.Query will be returned.

```
QueryParser queryParser = new QueryParser("keywords", new
  StopAnalyzer());
Query query = queryParser.parse("cat food");
```

Note: QueryParser is not thread-safe. A new instance of QueryParser should be created for each thread.

For this example the choice of query string is hard coded as "cat food". This query will result in all documents containing either "cat" or "food", but not necessarily both. It's possible to require that a document's keyword field contain "cat" and "food" when searching. Simply place a plus (+) sign in front of each word so that the search string will be "+cat +food" to require resulting documents to contain both "cat" and "food" in their keyword field. More advanced search options will be discussed later.

Next make a call to the Searcher's search() method, passing in the Query object.

```
Hits hits = searcher.search(query);
```

The search() method returns an instance of org.apache.lucene.search.Hits. The Hits class represents a collection of documents matching the search criteria, along with each document's relevancy score. These scores range from 0.0 to 1.0 where 1.0 is considered highly relevant and 0.0 is considered completely irrelevant (and not included in the Hits collection).

Finally, cycle through each Document returned in the Hits object displaying the SKU and name of the product along with its relevancy score.

```
for (int i = 0; i < hits.length(); i++) {
  Document document = hits.doc(i);
  float score = hits.score(i);
          System.out.println(document.get("sku") + " :: "
+
                document.get("name") + " :: " + score);
       }
```

sku	name	price	keywords
CAT-1234	8lb Kitten Food	8.99	kitten cat food
CAT-1235	20lb Cat	14.99	cat food
PC-1212	Small Pet	29.99	dog cat kennel
PC-1213	Large Pet	39.99	dog cat kennel
CAT-2211	16oz Catnip	4.99	cat catnip
CAT-3322	Cat Toy	12.99	cat toys
DOG-1199	42oz Water	10.99	dog food bowl
BRD-9212	40lb Large	34.99	bird parrot food

TABLE 2 QueryParser syntax

SEARCH COMPONENTS

A Searcher (org.apache.lucene.search.Searcher) is used to access a Lucene index and query its contents. There are two subclasses of Searcher: IndexSearcher that searches a single index and MultiSearcher that searches one or more indexes and collects all the results in a single result set.

Searches are performed by calling one of Searcher's search() methods and passing it a query (org.apache.lucene.search.Query). The search method returns an instance of org.apache.lucene.search.Hits. The Hits class is an array-like collection of documents that matches your query. The documents are ordered in Hits by a relevancy score.

A Query object can be constructed using org.apache.lucene.queryParser.QueryParser. QueryParser's parse() method parses a query string that's written in its query language and builds an appropriate Query object for that query string. QueryParser also uses an Analyzer in performing the parsing of the query string. It's not required, but it is strongly recommended that you use the same Analyzer for parsing queries that you used when indexing your documents.

Advanced Queries

Up until now, the queries have been relatively simple ones such as "cat food" and "+cat +food". QueryParser has a powerful selection of query operators to facilitate more complex searches. Table 2 lists all of QueryParser's operators.

Wildcard queries are fairly straightforward. The "*" operator can be replaced by zero or more characters to match a word. The "?" operator is replaced by exactly one character when matching. For example, "ca*" will match "cat", "car", "cap", or "candle", while "ca?" will match "cat", "car", and "cap", but not "candle". This is consistent with the behavior of "*" and "?" on a DOS or Unix command line.

The tilde (~) character, when used alone, performs a fuzzy search, matching words that are spelled similarly. For example, "cat~" will match "cat", but it will also match "car" and "rat" because these words are similarly spelled.

Surrounding two or more words with quotes (" ") produces a phrase. When two or more words are part of a phrase, those words must appear together in order to be considered a match. For example, ""dog food"" will match documents where "dog" is immediately followed by "food".

If a tilde and a number follow a phrase, then a proximity search is performed. For example, ""dog food"~10" will produce results where "dog" and "food" are found within 10 words of each other, but not necessarily adjacent to each other.

The carat (^) is a term booster. What this means is that any word followed by a carat is considered to have higher relevance than words not followed by a carat. For example, "dog^ kennel" will match where the document contains "dog" or "kennel", but will give a higher relevance to documents containing "dog".

The Boolean operators, AND, OR, and NOT behave as you would expect them to. For example, "(cat AND food) OR bird" returns all documents containing "cat" and "food" along with all documents that contain "bird". "cat NOT food" returns all documents containing "cat", but not containing "food". As you have seen before in the simple "cat food" example, OR is the default conjunction operator.

As shown in the previous example, parentheses can be used to group terms into subqueries.

As discussed, the plus sign (+) requires that a word or phrase exist in a field. Conversely, the minus sign (-) prohibits a word from appearing in the results and is roughly equivalent to NOT. For example, "dog –food" returns all documents containing "dog" but not containing "food".

Finally, there are times when you may want to search multiple fields. When constructing a QueryParser, you must specify a default field to be searched upon. Unless you specify otherwise, any words in your query will be looked for in the default field. In the examples, "keywords" is the default field. You can search on nondefault fields (assuming that they're indexed) by using a colon (:). For example, had the name field been tokenized and indexed, the query string "+cat +name:nummies" would return all documents in which the keywords field contains "cat" and the name field contains "nummies".

Crystal Reports 9 arrives!

Java developers rejoice over the introduction of a Java SDK featured in the industry-leading Crystal Reports 9.

Reporting software wakes up and smells the Java

SAN FRANCISCO – Delighted Java developers flooded the streets yesterday after experiencing the powerful new Java features of Crystal Reports 9®. The industry-leading software now offers a Java SDK for creation and integration of reports into J2EE-based applications.

HIGHLIGHTS

- Component reuse with single point updating
- Support for Java®, .NET and COM
- Unlimited SQL control
- New scalable report server for web application integration
- Flexible deployment licensing for enterprise web applications

See PORTAL & WIRELESS INTEGRATION page A2

Why didn't someone think of this before?

Crystal Reports 9 is now available with flexible, cost-effective deployment licensing designed for enterprise web applications. [illegible] sources say. that Crystal Reports request queuing capabilities are ideal for deployment of web applications with variable user loads.

Component reuse saves time, effort

Certified Training straight from the source

NEW YORK – Crystal Decisions™, makers of Crystal Reports, offers a full range of online, public and onsite certified training programs. The industry - lead[illegible] now offe[illegible]

To learn more, visit www.crystaldecisions.com/cr9/002/ and sign up for a free online seminar. To purchase, call 1 888 333 6007, go to our online store at www.crystaldecisions.com/cr9/023/, or contact your preferred reseller.

Access. Analyze. Report. Share.™

crystal decisions™

Customizing Lucene

While Lucene comes with an impressive set of functionality, you may still find that you want it to do something more or different than is available out of the box. As a search framework, Lucene provides several hooks for you to extend and/or modify its behavior.

In the previous examples, the analyzer chosen was StopAnalyzer. Underneath the covers, StopAnalyzer uses LetterTokenizer to tokenize text into individual words. LetterTokenizer treats any nonalphabetic character as a delimiter. This is fine in most cases, but what if you want to tokenize text that contains numeric characters ("0" – "9") as well as alphabetic characters? This would be desirable if the keyword text contains part numbers or model numbers. LetterTokenizer wouldn't help in this case.

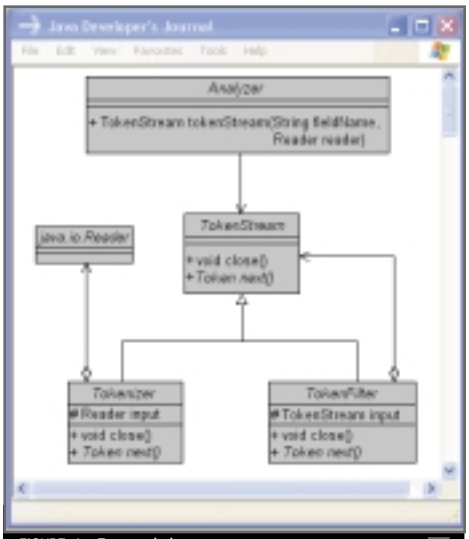


FIGURE 1 Text analysis components

Listing 6 defines AlphanumericTokenizer, a tokenizer that works like LetterTokenizer except for one small difference: it treats numeric characters as token characters along with alphabetic characters. It does this by subclassing LetterTokenizer and overriding the isTokenChar() method to return the results of LetterTokenizer's isTokenChar() implementation OR'd with a call to Character.isDigit().

AlphanumStopAnalyzer (see Listing 7) is an analyzer that uses AlphanumericTokenizer. The stop-word behavior of StopAnalyzer is still desired, so AlphanumericTokenizer is wrapped with a StopFilter. To normalize the text to lowercase, StopFilter is then wrapped with LowercaseFilter. AlphanumStopAnalyzer is functionally equivalent to StopAnalyzer, except, since it uses AlphanumericTokenizer, it does not treat numeric characters as delimiters. To try out AlphanumStopAnalyzer, use it in place of StopAnalyzer in both ProductIndexer and ProductSearcher. Be sure to reindex with ProductIndexer before searching the index with the new analyzer.

Suppose that synonym-matching capability is required so that "cat" will match "kitten", "kitty", or "feline". AliasFilter (see Listing 8) is a subclass of TokenFilter that does this. AliasFilter retrieves its synonym list from entries in AliasFilter.properties. For example:

```
cat=feline kitten kitty
dog=canine puppy mutt
food=feed chow
parrot=bird
```

With each invocation of next(), AliasFilter first checks to see if there are any synonyms in the alias stack. If there are, it pops the next alias off the stack and returns it. Otherwise, AliasFilter retrieves the next token from the input TokenStream, adds any aliases that may exist to the alias stack, and then returns the next token.

AliasAnalyzer (see Listing 9) constructs a TokenStream that does everything the TokenStream from AlphanumStopAnalyzer does, but it also uses AliasFilter to add synonyms to the TokenStream. To try AliasAnalyzer, use it as your analyzer instead of StopAnalyzer in both ProductIndexer and ProductSearch. Again, be sure to reindex before searching.

When trying AliasFilter you may discover some strange, albeit desirable, behavior. Search for "feline". Even though there are no aliases for feline, all cat-related products appear in the search results. Why? When you use AliasAnalyzer to search for "feline", the token stream does not expand beyond "feline". So why do "cat" products appear? The reason is, you also used AliasAnalyzer to index the products. When you indexed a product containing "cat", AliasAnalyzer expanded the token stream to include "kitten", "kitty", and "feline" in the index. When searching for "feline" it will be found in products whose token stream was expanded to include "feline". In effect, you get an automatic two-way aliasing between "cat" and "feline", even though it appears to be only one way in AliasFilter.properties.

Another common problem in searching is paging the results. A search query could return anywhere from zero results to a seemingly infinite number of result documents. Good usability practices suggest that you page the results, showing the user only a handful at a time. This can be accomplished in Lucene using result filters.

To create a result filter, you must subclass org.apache.lucene.search.Filter. The only required method is the

TEXT ANALYSIS COMPONENTS

When a field is tokenized, its content is broken into one or more tokens or words. Facilitating this tokenization process is the notion of an analyzer (see Figure 1). An analyzer is any subclass of org.apache.lucene.analysis.Analyzer that defines the rules for tokenization.

A token stream is an iterator that returns the next token with each call to its next() method or returns a null when there are no more tokens in the stream. Two important subclasses of TokenStream are Tokenizer and TokenFilter. Both of these classes are abstract and must be subclassed to define the specific rules on how to tokenize content.

At the core of the tokenization process is a Tokenizer. A Tokenizer wraps an instance of java.io.Reader and performs the actual work of breaking a stream into individual tokens (not unlike the notion of a StringTokenizer).

TokenFilters act as decorators of other TokenStreams. Token filters can be used to add, replace, or remove tokens from a TokenStream. For example, org.apache.lucene.analysis.PorterStemFilter is a TokenFilter that replaces each word in a TokenStream with its word stem (e.g., "painting" becomes "paint").

Analyzers rely on token streams (subclasses of org.apache.lucene.analysis.TokenStream) in defining the tokenization rules. In fact, an analyzer is nothing more than a factory for creating instances of TokenStream.

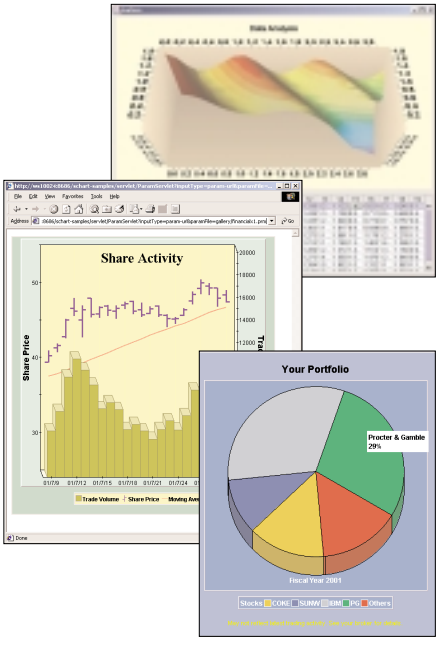
To see how the text analysis components are used together, consider some of the TokenStream and Analyzer implementations packaged with Lucene. StopAnalyzer is an analyzer whose job is to remove stop words (e.g., "and", "or", "the", etc.) from a tokenized stream. At the core of StopAnalyzer is an instance of LowerCaseTokenizer. It tokenizes the stream into individual words, normalizing them to lowercase as it goes, where any nonalphabetic character is considered a delimiter. An instance of StopFilter decorates LowerCaseTokenizer, removing stop words from the stream as they're found. StopAnalyzer's tokenStream() method is merely a factory method that returns the decorator chain made up of LowerCaseTokenizer and StopFilter.

sitraka
JClass®

The ultimate Java™ server and client components

sitraka
JClass®

Sitraka JClass is the only professional-quality Java component family that covers every facet of Java development – sophisticated client-side user interface components, reporting and utility components, server-side web client components, and web services enabled components. With unrivaled platform, JDK, web/application server and IDE support, JClass works everywhere your application does, or will.



Sitraka JClass DesktopViews

The only components you need for Java applications and applets: 2D/3D charts, tables, reporting, data-entry fields, database, tree view, layout managers, and much more. Build enterprise-class user interfaces quickly and cost-effectively.

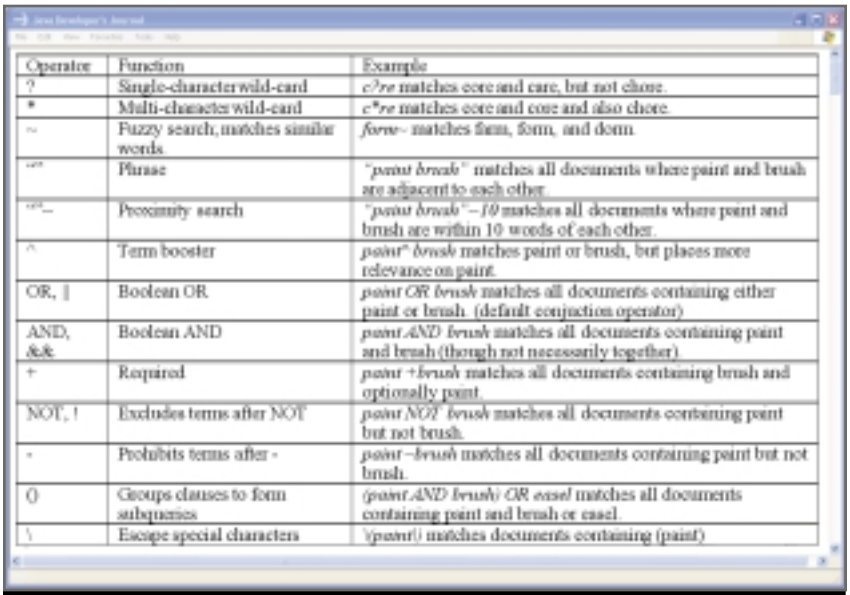
Sitraka JClass ServerViews

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

Evaluate and experience JClass for yourself – visit:

www.sitraka.com/jclass/jdj

sitraka
the Java™ advantage



Operator	Function	Example
?	Single-character wild-card	*c?re* matches core and care, but not chore.
*	Multi-character wild-card	*c*re* matches core and core and also chore.
~	Fuzzy search, matches similar words	*form~* matches form, form, and dorm.
""	Phrase	*"paint brush"* matches all documents where paint and brush are adjacent to each other.
""~	Proximity search	*"paint brush"~10* matches all documents where paint and brush are within 10 words of each other.
^	Term booster	*paint^ brush* matches paint or brush, but places more relevance on paint.
OR, ||	Boolean OR	*paint OR brush* matches all documents containing either paint or brush. (default conjuction operator)
AND, &&	Boolean AND	*paint AND brush* matches all documents containing paint and brush (though not necessarily together).
+	Required	*paint +brush* matches all documents containing brush and optionally paint.
NOT, !	Excludes terms after NOT	*paint NOT brush* matches all documents containing paint but not brush.
-	Prohibits terms after -	*paint -brush* matches all documents containing paint but not brush.
()	Groups clauses to form subqueries	*(paint AND brush) OR easel* matches all documents containing paint and brush or easel.
\	Escape special characters	*\(paint\)* matches documents containing (paint)

TABLE 3 Field types and field static factory methods

bits() method. It will return a java.util.BitSet where each bit represents a document in the result set. If the bit is true, the document will be returned in Hits, otherwise it won't be returned.

PageFilter (see Listing 10) is an example of a Filter that's used to paginate search results. Given a page number and a page size, PageFilter will pare down Lucene's result set to a specific page's subset of documents. It does this by creating a BitSet big enough to hold the maximum number of result bits and then looping through the bits that need to be turned on. To use PageFilter, change ProductSearcher's call to search() to look like this:

```
Hits hits = searcher.search(query,new
PageFilter(1,20));
```

This new call to search() will result in showing only the second set of 20 results.

Conclusion

Building a full-featured search engine can be a daunting task. But, thanks to Lucene, much of the complicated details are abstracted behind an easy-to-use API. We've seen how easy it can be to create an index for searching practically any type of information. We've also seen how Lucene is flexible and can be extended to satisfy custom indexing and searching requirements.

Resources

- *Jakarta Lucene:* http://jakarta.apache.org/lucene
- *NLucene, the .NET implementation of Lucene at SourceForge:* http://sourceforge.net/projects/nlucene
- *JGuru FAQ on Lucene:* www.jguru.com/faq/Lucene
- *About Lucene's creator, Doug Cutting:* http://lucene.sourceforge.net/background.html

Author Bio

Craig Walls is the manager of Internet development for a Dallas, Texas-based retailer. He has eight years of experience in software development, six in Java. Craig is a Sun Certified Java programmer and a Sun Certified architect for the Java platform. He holds a BS in computer science from New Mexico State University.

javaman@habuma.com

Application integration is an intimidating challenge for any enterprise; the downtime costs of unreliable integration are unthinkably expensive. To keep profits climbing, enterprises are adopting open standards-based messaging, caching and integration solutions.

The SpiritSoft framework integrates your legacy applications into a single JMS-compliant environment. SpiritSoft technology goes beyond JMS to give you a secure, scalable and robust enterprise-integration strategy to provide for your future messaging and Web Services requirements.

Secure your integration foothold … there's no room for error.
Download our white paper www.spiritsoft.com/climber

spiritsoft.
go beyond jms

Webtop Architectures for J2EE

Webtops explained

WRITTEN BY
PATRICK SMITH

In the early '90s, many companies invested in traditional client/server architectures by building fat-client applications with rich graphics that offloaded legacy-system processing time.

GUI business applications presented lower runtime costs than the CICS applications they replaced due to reduced demand on the mainframe server. This fat client would handle all business logic and data validation and then commit data to the server.

The emergence of Java and the Web, however, prompted another major shift in thinking about building applications. With Internet-based technology, suddenly companies could extend their reach by delivering dynamic HTML in a browser to anyone with a Web connection. This new Web-enabled world thus reversed the earlier trend, and application development began moving business logic back to the server.

Many early adopters among the Fortune 500 migrated their legacy applications to a J2EE architecture, attracted to this emerging paradigm by the ease of deployment afforded by browser-based delivery of the presentation layer. For a significant number of these companies, however, the strategy backfired, as they increasingly ran up against the usability limitations of HTML, particularly for more complex GUI workflow, business calculations, or sophisticated graphics. What's more, browser-based front ends introduced performance inefficiencies as the application must make round-trips to the server to paint new views, a process that can be cumbersome because of the browser's inherent limitations in windowing and real-time graphics capabilities.

Today, Web services offers the promise of allowing Internet-based applications to interrelate in new ways, giving businesses a powerful new tool for collaboration. But a browser-based presentation strategy forces architects of these applications to make an unpleasant trade-off:

- Either drastically simplify the presentation layer by distributing content over a greater number of screens, or
- Meet the business requirements and deliver markedly slower performance, lower usability, and diminished functionality compared to the original mainframe applications that the new J2EE applications are designed to replace.

How can we deliver a new generation of user interfaces that enable us to leverage the cooperative power of Web services? As B2B becomes more mainstream, the network becomes the backbone of transactions, both within and among the enterprises.

The Webtop Solution

An emerging solution to this problem is the Webtop. The Webtop is a next-generation rich-client user interface, a hybrid Swing-based GUI front end that connects to the J2EE application server over the Internet (see Figure 1). Webtops bring the best of old-style client/server to the J2EE age, enabling rich end-user functionality that interacts with a remote server. Using technologies such as SOAP, the Webtop is responsible primarily for the presentation layer, relying on the application server for business logic and distributed transaction processing.

The Webtop represents a return to the *n*-tier distributed client/server model by offloading costly view management to the desktop. This distribution of power can significantly improve performance and the user experience. Not surprisingly, many companies, including DaimlerChrysler, Sabre, BMW, Amadeus, and Deutsche Bank, are revising their initial J2EE strategies to take advantage of this architecture.

Let's examine this architecture.

MVC and Presentation State Engines

Webtops are based on the Model-View-Controller pattern familiar to Java Swing developers. In this pattern, the model is an object representation of the business, the view is the canvas, and the controller sits behind the view to control the navigation between individual graphic widgets and value holders on the view (see Figure 2).

But Webtop client/server applications have complex user interfaces, and state is a key consideration. Therefore, immediately below the Swing UI is a navigation state engine known as a navigation controller. Less well known to Swing developers, it's responsible for the application workflow control of windows that run in the Webtop. Use of a navigation controller reduces coupling in the Webtop application since individual view controllers are not required to have knowledge of each other. This approach also allows views to be interchanged easily as new versions are rolled out, and allows you to alter GUI behavior based on user roles, an architectural feature that's highly desirable for applications that are in a constant state of requirements fluctuation.

Architects of these complex Java applications define navigation controllers so they operate in four directions (see Figure 3):

- ***Up:*** To signal a view controller to open its view

- ***Left or right:*** To signal collaborating navigation controllers to take control
- ***Down:*** To request service from the application server (such as commit a transaction or request new objects)

The Webtop can be assembled with a single or multiple navigation controllers. When it's started, a primary navigation controller is instantiated, which then determines the state of the Webtop based on the last usage. Then the navigation controller messages the correct view controller and the Webtop picks up where the user left off. When exiting any particular view (from the view controller), the navigation controller must signal another controller to pop a view or it will request that the application server perform a transaction service.

There are several ways to implement this. The most common approaches are to either use a single navigation controller instance that has knowledge of all the GUIs or, for more complex applications, employ a series of navigation controllers, one for each Swing user interface. In this scenario, each Swing client uses an association manager with event listeners to determine who to pass control to when exiting a specific view. When using multiple navigation controllers, a central association manager mechanism can be used for registration.

Communication between the Webtop (outside the firewall) and the application server can be accomplished using the Simple Object Access Protocol (SOAP). Use of SOAP provides remote method invocation over HTTP. Because the Internet protocol does not guarantee delivery, architects should consider the latency of the production network on which the Webtop will communicate with the server. If latency or uptime is an issue, consider implementing SOAP with asynchronous messaging versus the more popular synchronous messaging. Implement an outgoing message type that is sent to the application server paired with an opposite message response to indicate that the transaction was successful. This type of messaging is common in telephony network management where servers are used for node provisioning, a remarkably similar model to Webtop management.

Application Management for Webtops

Companies that follow the Extreme Programming (XP) model benefit from the ability to revise applications rapidly as user requirements continue to change and evolve. The XP model involves iterating more frequently, producing a significant number of application updates. In the Webtop environment, we know we have to update the server and the client, a client that connects over the Internet.

Consider a Webtop environment for travel services automation deployed to thousands of independent travel agencies in North America. It's unlikely that travel agents in various geographic locations running on a Webtop would be expected to download new installer-packaged versions of that Webtop weekly – updates corresponding to server-side changes. Moreover, how would we deal with application failures that may be associated with a version changeover?

The browser experience offers an immediacy and seamlessness that can be emulated in the Webtop environment. When the user double-clicks a desktop icon to launch the Webtop application, he or she should be presented with the most current functionality available. The travel agent should open the Webtop and expect to have access to the current functionality on the server.

Deployment and Management Challenges of Webtops

The Webtop architecture, while powerful, introduces new challenges for companies that are adopting this approach:

- ***Webtop version management:*** Centralized control over who is using what.
- ***Fault tolerance and monitoring:*** In the event of a critical failure of a 24x7 application, administrators must be notified automatically and centralized control must provide immediate rollback capability.
- ***User authentication and access control:*** Centralized control of user authorization and access to the Webtop version.
- ***Reporting:*** The more distributed applications are in production, the more important reporting becomes to provide information such as which users are running each version, how often, and when was the last connection time.
- ***JAR-level differencing:*** Webtops are powerful but have a large footprint due to the size of the JAR files. The JAR is what changes most, so companies have sought out updating technolo-

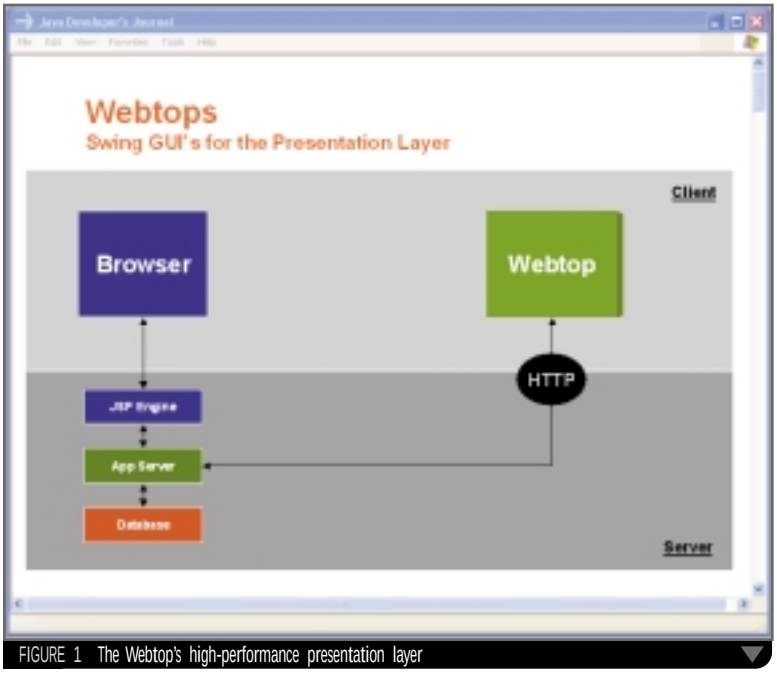


FIGURE 1 The Webtop's high-performance presentation layer

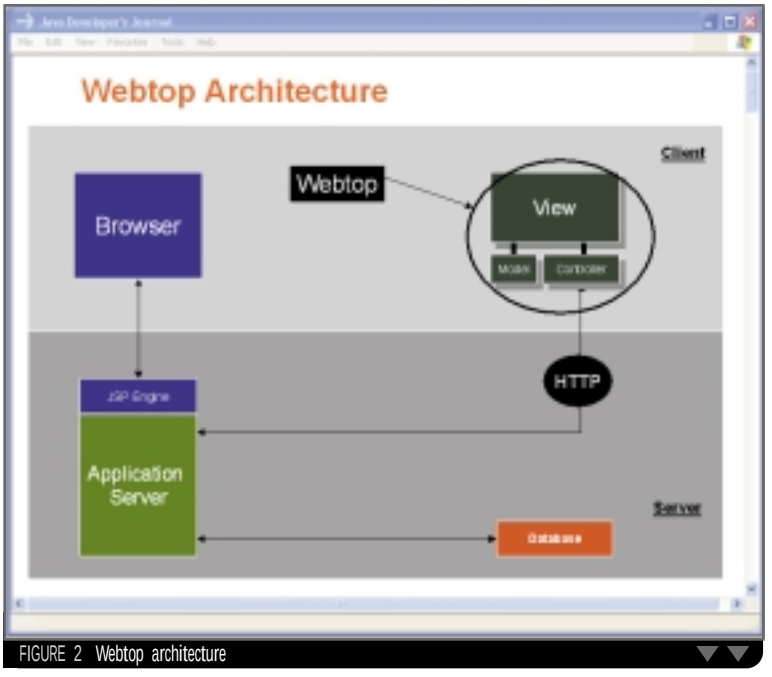
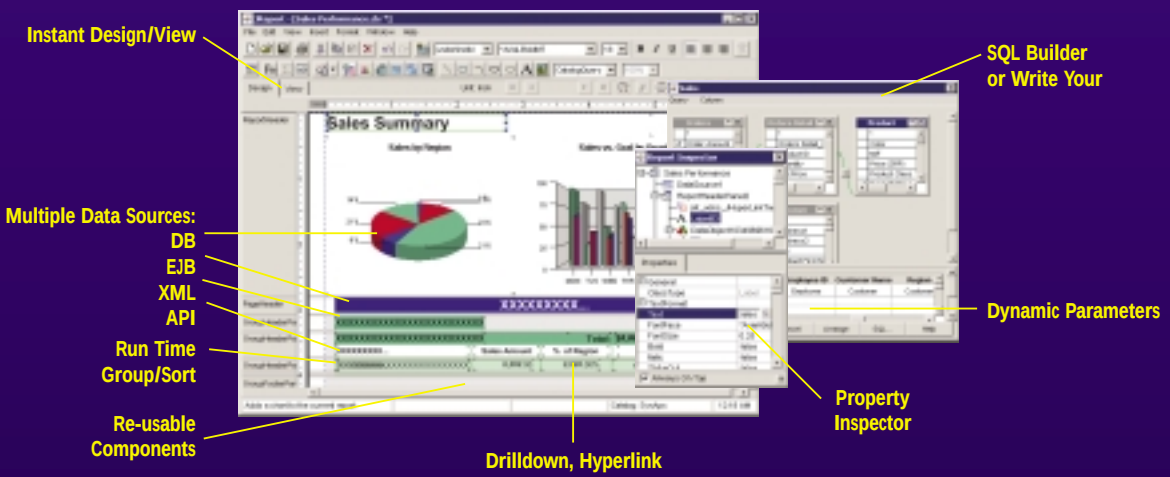
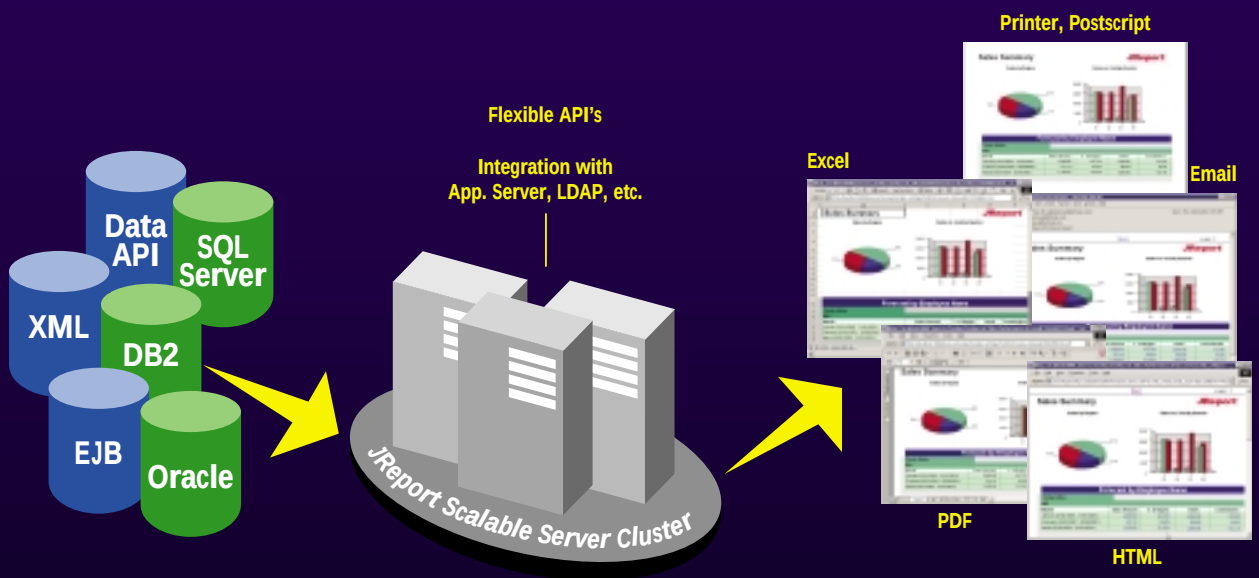


FIGURE 2 Webtop architecture

DESIGN ONCE...



REPORT ANYWHERE.



JREPORT™

JReport lets you take your reports anywhere you want. Imagine creating reports that pull together data from virtually anywhere, arranging them any way you wish, and delivering them precisely – whether in pure HTML, PDF, Excel, XML, JSP, email, or your network printer. Imagine having ad hoc reporting ability to create or modify reports on the fly, right from the end user's browser.

And all this is built on 100% J2EE architecture, to provide secure, reliable, scalable performance and easy integration with your enterprise systems. You can count on JReport: any report, any format … anywhere.

JINFONET SOFTWARE

FREE DOWNLOAD! VISIT www.jinfonet.com/j12.htm OR CALL 301-838-5560

gies that can compute the difference between JAR files from different versions of a Webtop application.

- ***JRE management:*** Guarantee that the correct JRE version is associated with the version of the Webtop on the desktop; there can be no dependency on the user to ensure the JRE is present or correct.

Application Management

The Internet is a pull-based system, and so application management platforms for Webtops must therefore be pull based. The emerging trend, proven by Fortune 100 early adopters, is to deploy software agents attached to the Webtop instance. Software agents provide remote-control capability since they execute according to a centralized policy server or deployment server cluster. This client-side agent administrator connects to the cluster to confirm application updates, user access, and authorization policies (see Figure 4). The agent will report information about the Webtop usage and health, including an immediate alert should the agent detect a critical failure. A centralized control console is usually included with a commercial pull-based deployment platform.

How is this performed? Software agents are typically implemented as headless lightweight software applications that perform tasks via remote control. The communication pattern between the agent and the server is sometimes called double dispatching. When the client agent is instantiated, it doesn't know what its behavior should be, even though there are obviously methods implemented on the agent. The agent connects to a server, describes itself, and requests instructions or tasks to perform. The server replies by providing a policy set that describes what the agent must do, which in reality is a subset of behavior the agent always had.

Webtop client agents operate similarly – they're programmed to download a rule set from a central server and execute the rules based on a time schedule specified in the rules. Rules or policies can be implemented in plain text files or as XML. The policy is specified using a standard protocol for network launching.

The most important aspect of the agent model for Webtops is its ability to dynamically update the Java classes and the JRE on the fly, as part of a version update. To accomplish this, the agent is configured as a wrapper for the Webtop, so what appears on the desktop as an icon is actually the agent and not the real Webtop. When the user invokes the Webtop, the agent is invoked and must then examine its policy on file for the Webtop. Each time the agent is in communication with the deployment server, it may receive updated policy information such as when updates should be performed and whether or not these updates are mandatory.

In the travel agency use case, the client-side agent would be configured by the deployment server for mandatory updating on startup of the Webtop. The Webtop agent would connect to the server over HTTP to confirm that the version of the Webtop running on the desktop is the correct version specified by the application administrator at the central location. If a new version has been placed into production, the deployment server is responsible for computing the class differences in all JAR files, plus the differences in files such as help files, graphics, and so on. The set of differences is cached on the server.

When connecting to the server, the agent must perform the following:

1. Verify that the user is authorized to access the Webtop and confirm that the version matches the production version specified as current on the server.
2. Download the set of updated class files in the JARs cached on the server and any additional files that must be updated, such as icons, graphics, and help files. Since the JAR differences are significantly smaller than the actual Webtop JARs, this procedure is performed quickly, even in low bandwidth conditions.
3. Open the corresponding JAR files of the local Webtop, update the classes that have changed, and re-JAR the files.
4. Confirm and/or download an alternate JRE if specified for the new Webtop version (this is specified by the application administrator).
5. Remove files that are no longer required for the alternate version.
6. Invoke the main start-up class for the Webtop and start the application.

What happens when a recently updated Webtop then fails in produc-

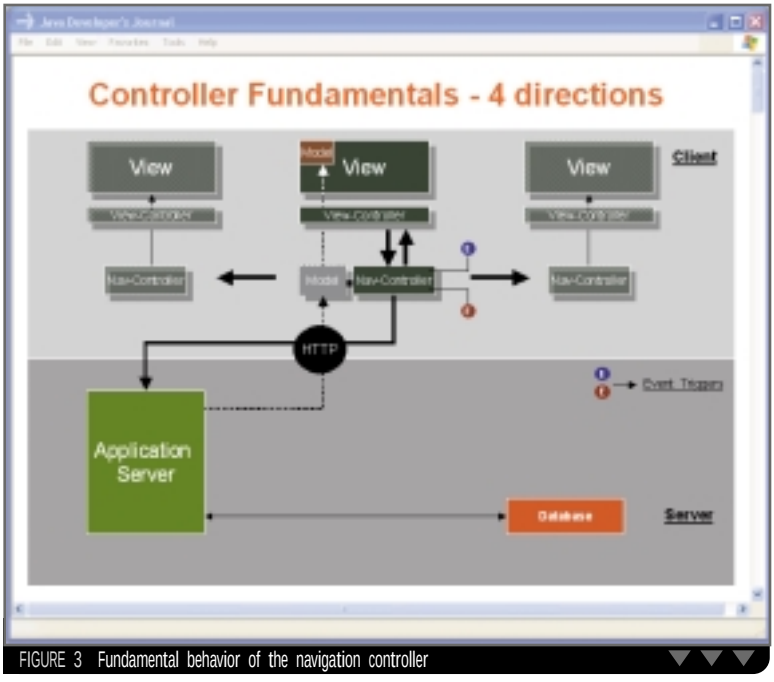


FIGURE 3 Fundamental behavior of the navigation controller

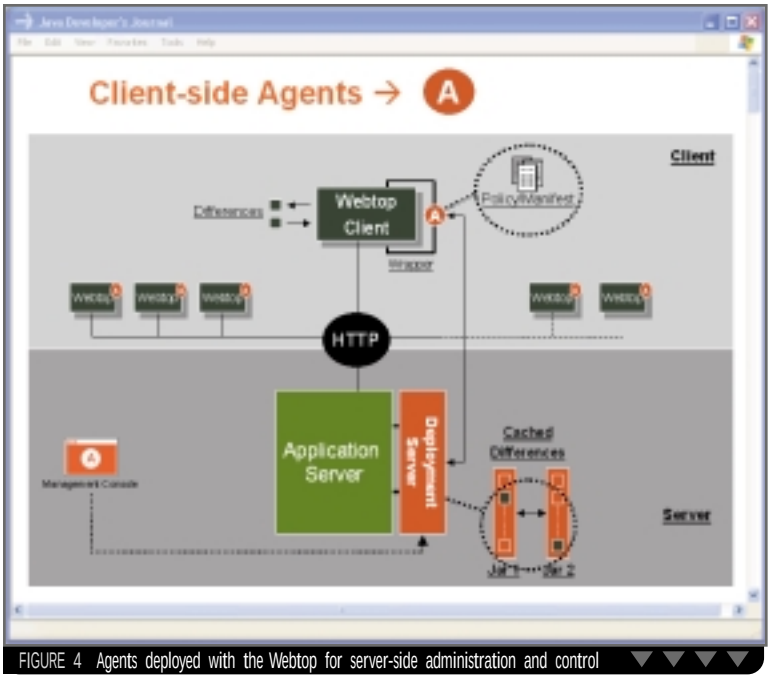


FIGURE 4 Agents deployed with the Webtop for server-side administration and control

IBM

WebSphere® software

OPEN TOOLS PLAY

1] WIN WITH OPEN TOOLS: The CIO doesn't want excuses. That's why you need software that lets you combine your favorite development tools into an integrated suite.

2] WIN WITH WEBSPHERE: IBM WebSphere Studio automates tasks so you can use your preferred toolsets in a single, integrated application server–neutral environment. WebSphere. Part of our software portfolio including DB2®, Lotus® and Tivoli.®

3] MAKE THE PLAY: Visit **ibm.com**/websphere/opentools to download WebSphere Studio code today.

e business is the game. Play to win.™

> Pull-based deployment marks a significant advancement in Java application management

tion? At best this is unacceptable in a B2B environment – a source of embarrassment. At worst, failure can take its toll on productivity and revenue. One of the more interesting capabilities of agent-based administration is an agent's ability to detect errors by monitoring the standard Java error channel. Developers know that all application exceptions should be handled, but the realities of today's development environments may not always permit.

Requirements are a moving target, and the advent of XP means more updates are frequently rolled out to users. Naturally, the probability of unhandled exceptions increases with the number of releases. Unhandled exceptions are an unfortunate reality in production applications, and when they occur, the Webtop becomes unstable or does not work. The software agent, by detecting the error, can immediately issue an alert to the administrator. This alert allows the production monitoring staff to take corrective action such as a rollback to a previous, more stable edition (if necessary, in the middle of the night) until a repair can be made.

Pull-based deployment marks a significant advancement in Java application management. Rollback or new version rollout can now occur seamlessly and without user intervention. When users start the Webtop, they're insulated from on-the-fly, background modifications to the Webtop code. Companies that have a large number of production applications with Webtop front ends have the option of deploying a single agent to manage all Java Webtops or deploying one agent per Webtop. Using the multiple agent strategy is arguably safer. The agents are small and should a single agent fail, it doesn't render all Webtops inoperative.

Some people confuse application management for Webtops with the popular install-packaging technology. While some installers now include auto-updating features, typically these technologies do not employ remote control agents; rather, updates are user-initiated. Moreover, the installers were not designed to compute JAR differences at the Java class level, or provide version management in high-availability production environments (as such, installers are a better fit for ISVs that update standalone applications periodically). These are critical features in Webtop management and serve as primary drivers for corporations moving forward with Webtops and application management platforms.

References

- Gamma, E., et al. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Brown, K., and Shi, L. "Using the Distributed ICM architecture to achieve client-server independence." http://members.aol.com/kgb1001001/Articles/D-ICM/d-icm.htm
- *Java Web Start:* http://java.sun.com/products/javawebstart
- *DeployDirector (part of Quest Software):* www.sitraka.com/software/deploydirector

Author Bio

Patrick Smith is director of DeployDirector product management at Sitraka (now part of Quest Software). With more than 10 years of software development experience, he has extensive expertise in the development and deployment of global software applications employing distributed object technology and n-tier architectures.

patrick.smith@sitraka.com

DARE TO COMPARE?

RIP OUT THIS FEATURE LIST AND SEE HOW YOUR SERVER STACKS UP AGAINST MACROMEDIA® JRUN™ 4 SERVER

NEW AND ENHANCED FEATURES OF JRUN 4:	JRUN 4
Certified J2EE 1.3 Compatible	✓
High performance web container with JSP 1.2 and servlet 2.3 support	✓
Advanced EJB 2.0 support including local interfaces, message-driven beans, EJB-QL, and CMP 2.0	✓
Additional Java® 2 APIs -- JMS 1.0.2, JTA 1.0, JAAS 1.0, JCA 1.0, JAXP 1.1,JavaMail 1.1, JDBC 2.0, JNDI 1.2, RMI/IIOP 1.0	✓
Support for XML, SOAP, WSDL, and UDDI for web services	✓
Built-in production quality HTTP server	✓
Hot modification and auto deployment of applications	✓
Integrated XDoclet support for simplified component deployment	✓
Built-in Macromedia® Flash® MX Remoting services for high impact user interfaces	✓
Type IV JDBC drivers included for Oracle®, MS SQL Server, IBM® DB2, Sybase®, and Informix®	✓
Highly customizable and extensible JMX-based service-based architecture	✓
JMX-enabled web-based management console	✓
Advanced multi-tier server clustering enabled by Jini® technology.	✓
Leading OS platforms, including Unix, Linux®, and Windows®	✓
FULL-FEATURED PRODUCTION DEPLOYMENT	**$899/CPU**

Java Compatible™
Enterprise Edition

We've packed JRun 4 with features to make it faster, easier, or more affordable than any other J2EE server you can find. Don't just trust what you read, try it out yourself. You'll see why over 10,000 leading organizations deploy their J2EE applications on JRun.

Our FREE non-expiring developer version will get you started. Download JRun 4 today for FREE at:

www.macromedia.com/go/jrun4jdj

Go ahead....we dare you.

macromedia®
JRUN™4

JASON BELL J2SE EDITOR

Supporting Other Industries

Summer 1999: I was fortunate enough to be working on a Java project to write an Internet airline ticket reservation system for a client. It was one of my first Java programs and many a sleepless night was had. It opened my eyes to the travel industry and how the International Air Transportation Authority (IATA) used EDI data to communicate airline, hotel, and rail information to travel agents (it was also my baptism of the ways of StringTokenizer and Socket classes).

Fast forward to 2002: I was reading some of the Java Community Process papers online and stumbled across the "Travel Industry Reservation Booking Foundation API Specification" initiated by Sherry Shavor at IBM. At first I was elated that someone was taking the travel industry seriously and wanted to create an API. I took another gulp of coffee (with a small amount of smug pride) and continued reading. When I went back to the top the of page I noticed one word that changed everything – in a big bold red font I saw the word "withdrawn."

Okay, I was deflated, but the withdrawal of the specification was due to the lack of community support, not industry support. The likes of British Airways, Open Travel Alliance, and Galileo International were in full support, so why can't we be?

Let me put it into perspective: Internet travel is one of the biggest areas in e-commerce and the turnover by 2004 is estimated to be over $30 billion. It seems to me the community has just turned down an incredible opportunity. Even if Java was accountable for 2% of that $30 billion, that's $600 million we've just turned down due to our lack of motivation to support an industry. Even the small airline carriers such as easyJet (a European airline modeled on Southwest Airlines' basic structure) adopted an Internet booking system. As of the summer of 2000, easyJet was handling 70% of its bookings through the Web. It was an easyJet employee who suggested to management that they have a small booking system on the Internet. From humble beginnings come mighty things.

It's all very well that the Java community is shouting from the sidelines, "You have to use our technology – it's great!" but then not embracing the needs of the industries that have the power and the money to adopt such technologies. The same people who stand on the sidelines are usually the ones screaming that .NET is inferior (that's a debate I steer clear of). At the end of the day it's attitude and humility that will win work for Java programmers and development teams. It's important for a client to know a team is passionate about the vision of the client instead of the single-minded vision of the programmer. The sooner we call ourselves servants to our clients the sooner we have a chance of making this whole thing work; we are all major links in the supply chain, not the end of it.

What is ahead for the future? Is there anything we should be watching out for? Well, I've been doing a lot of work on business intelligence using the Web as a big fact-finding engine. Companies are willing to pay to have these types of applications do large amounts of work that would take human intervention an age to complete. Think about the potential. If you have a hunch, go with it; what could possibly go wrong? Leave venture capital funding for the time being, until you're sure what you have is going to work. Anyway, VCs are so 1997.

The opportunities are out there; it's a change of mindset that's required from time to time. I'm certainly not saying we're all guilty, but a short pause to think about where we are and where we're going is always a good thing. I think the Pet Store demo benchmark fiasco was a wake-up call for some of us and it came at a welcome time. Yeah, we saw the cracks in the woodwork, but did we stop and think about it?

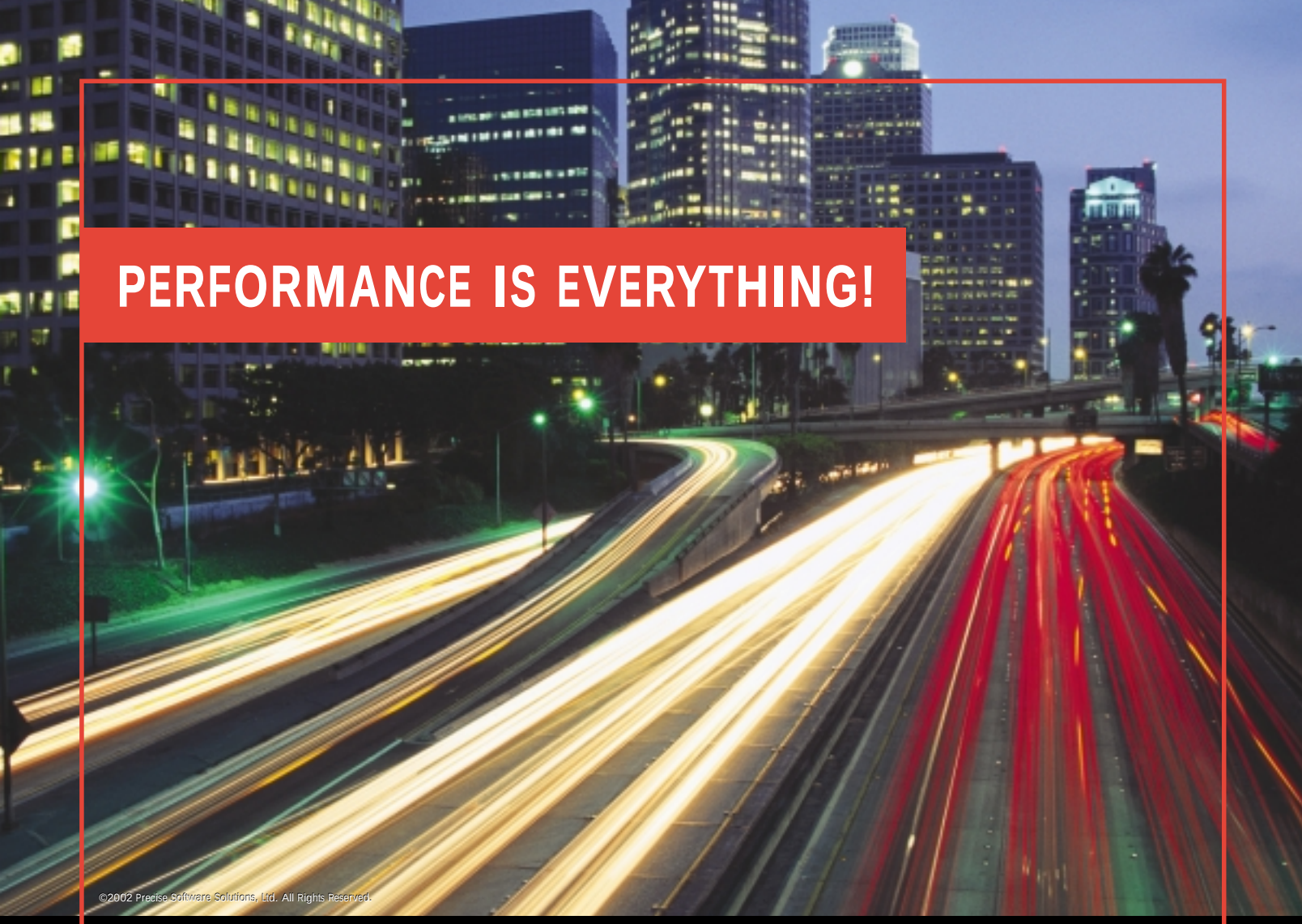
The longer we stand on the sidelines watching the world go by, the more time we give the competition to come along and steal Java's glory.

jasonbell@sys-con.com

AUTHOR BIO

Jason Bell is a programmer and chief technical officer for a B2B Web portal in York, England. He has been involved in numerous Web projects over the past five years, the last two of which have been servlet-based.

J2SE INDEX



Give your J2EE applications lightspeed performance.

To excel in today's business environment, your J2EE applications need to scream like a bat out of hell. With heavy traffic bottlenecks and crawling end user response times, you're total roadkill. Proactive detection of application performance problems is crucial to rapid resolution, and to taking the detour around a crisis up ahead.

Precise/Indepth for J2EE correlates web, J2EE and database server performance to get to the root cause of performance degradation, from URL to SQL and beyond™.

As Precise/Indepth for J2EE anticipates performance degradation, it instantly notifies your top mechanics for analysis and tuning deep within the middle tier of your web infrastructure and provides automatic drill-down analysis and recommendations. This translates into streamlined J2EE web applications and lightning fast response times.

Turn up the speed limit for your J2EE applications. Talk to your Precise account representative today at 1.800.310.4777, or visit **www.precise.com/jdj** to download a free white paper on J2EE performance management and to register for an upcoming Webinar on J2EE application performance.

Precise

Performance is Our Business

Plug in Your Command Processor Now

. . . and start saving money!

WRITTEN BY
RICHARD ROSS

This article details the implementation of a tool called the Command Processor. This tool takes a Java object and creates a command-line interface to its public methods.

These public methods are essentially your Application Programming Interface (API). During the course of this article we'll get a good look at the java.lang.reflect package and a chance to kick the tires on the Regular Expression package included in the 1.4 JDK.

I often find myself with fresh code and no convenient way to try it out. The GUI is not ready or there is no requirement for one. Even writing the argument processing for a main function is often far more work than it's worth. I want to be able to work with my code without modifying the API or writing a throwaway UI. In the long run, all the solutions I've tried were either too much work or required significant modifications to the class. Simply put, I want the Command Processor to create a command-line UI for any given Java class. Here are my requirements, in order of importance.

1. Allow execution of API functions from a command line with parameters
2. Allow the reuse of created objects (create and use variables)
3. Be completely decoupled from the API that it uses (no code modifications to the processed object)
4. Require minimal effort to utilize
5. Provide certain built-in functions (list, exit, etc.)
6. Restrict the API methods loaded

Requirements 1 and 2 allude to the basic UI that's created for any given class. I want to be able to type in a method call, then examine and reuse the returned object. For example:

```
Cmd:> myvar = createUser "John",
"\"fingers\" Doe", 'C', 34
Cmd:> myvar.toString
User: John "fingers" Doe, Age 34,
Clearance 'C'
Cmd:> othervar.addUser myvar
```

The entire syntax is described in the javadoc comments for the CommandProcessor class. It's very Java-like, but notice that parentheses are not required (they're actually not allowed) around the argument list. This makes it easier to handle argument casting later on. If you've ever tried writing a Lexer/Parser, you'll probably agree that parsing these lines by hand would be fairly difficult. Embedded quotes, for example, can cause a ton of grief to the programmer. Similar tools (DJava, for example) use a full-blown parser generator like Antlr or JavaCC to parse Java strings. I didn't need that level of sophistication, so I used the tools available to me in the 1.4 JDK. I chose to write a simple pseudo lookahead parser and found that the StreamTokenizer class gave me a good head start.

One of the keys to this tool (and requirement number 4) is that it must not be a burden to use. I want to create an instance of the Command Processor, hand it an object, and start working. As much as possible, I want to avoid configuration files and coded dependencies. Introspection comes to the rescue here as it allows us to examine the declared and inherited members (which include fields, constructors, and methods) of a Java class. It also allows us to fulfill requirement number 3, complete decoupling. As you'll see in the code API.java, the Command Processor and the class it processes (the processee, if you will) are completely unrelated. (The source code and Listings 1–5 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

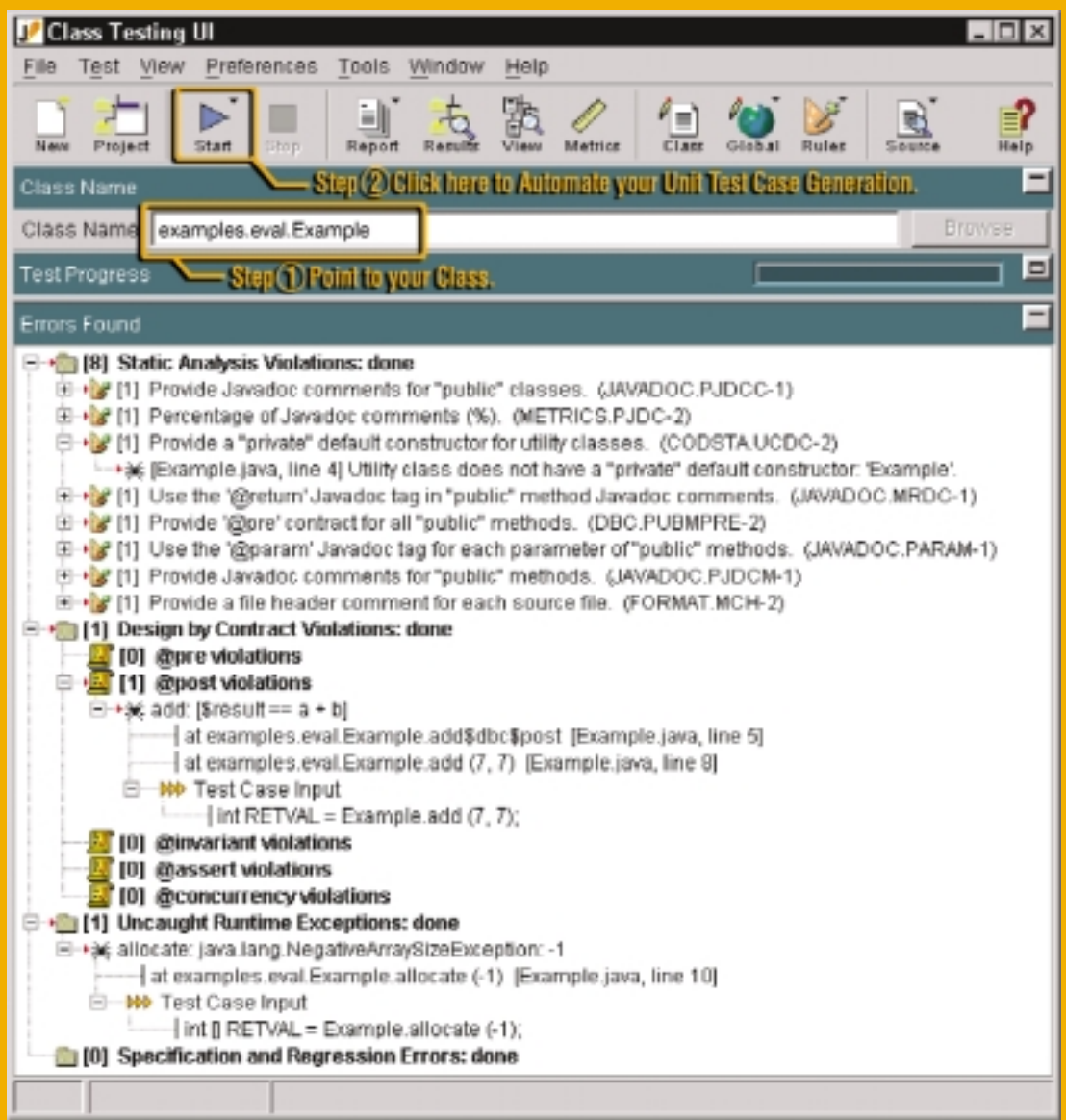
Tokenizing

Since the first thing I'll have to do is parse command lines, we might as well start there. First, a couple of rough definitions: Lexers (sometimes called Tokenizers) take a stream of characters and turn it into a stream of tokens. Parsers take this token stream and use them somehow. Tokens are collections of characters that have some semantic value in the language you're using. Tokens are essentially the words of our language. The token "_aFunction" could well be a Java identifier, while the token "23.3e7" is quite likely a number.

Starting with a given command-line string, I first break it into tokens. Even for our simple language, this quickly becomes a difficult task to do by hand. You end up with a giant decision tree and an awful lot of if/else and switch statements. However, lexical analysis is a well-understood field and Java provides a class that you can use. The java.io.StringTokenizer class takes some kind of Reader as its input and provides functions for retrieving tokens and setting various parameters. Essentially, all the if/else and switch statements are there. Most of them are in the 230-line nextToken() function! It's not without its quirks, but since James Gosling is listed as the original author, I'll just assume I didn't fully understand it.

Running the code in StreamTokTest.java will give you a good idea of how useful the StreamTokenizer is right off the bat (the output is shown in Listing

Automated Unit Test Case Generation for Java™ with Parasoft Jtest®



Features

Supports JUnit test[1] cases and enhances JUnit capabilities.

Automatically checks more than 300 Java coding guidelines.

Provides framework for adding custom guidelines.

Automates the creation of new test cases.

Platforms

Linux
Windows NT/2000/XP
Solaris 2.7 and higher

Parasoft Jtest reads and analyzes code, creates harnesses, stubs and test inputs to test the code without user intervention. In short, Jtest is the most comprehensive, automated and easy-to-use testing tool in the industry.

Jtest enables you to perform regression testing and static analysis.

Jtest also supports existing JUnit test cases and automates the creation of new JUnit-compatible test cases.

PARASOFT
AUTOMATED ERROR
PREVENTION

Parasoft Jtest is part of Parasoft's Automated Error Prevention Solution for Java™.

Download an evaluation copy at http://www.parasoft.com/jdj12.
Or call a Parasoft Error Prevention Specialist at (888) 305-0041.

PARASOFT jtest®

1). It handles comments, quoted strings with nested quotes, and chars in single quotes without even setting a parameter. It's not a perfect Java Lexer out of the box, but it wasn't meant to be. The tokens returned by StreamTokenizer have three properties: type, sval, and nval. Type is an int that represents the predefined type found or the single non-whitespace character that follows the last token. sval and nval are string and double representations, respectively, of the token and are only valid if the token type was TT_WORD or TT_NUMBER. Notice that the valid Java double 2.01e3 was broken into two tokens, the number 2.01 and the word e3.

There are three other things worth noting: a word and a quoted string have distinct types, the single quote is treated the same as the double quote, and the numeric value is not erased between tokens. However, the only thing that really matters to me is that Java numbers don't all parse correctly.

How do I tell the Tokenizer that 2.01e3 is a number or 2L for that matter? I can't. The Tokenizer just keeps adding characters to the token until it finds a character that's not in the current token type. It doesn't know anything about context, so we'll put off numeric identification until we get to the parser. To do that, I'll need the Tokenizer to treat numbers the same way it treats letters. Strangely, the Tokenizer won't let me unset the numeric attributes it has set. I have to clear everything with a call to resetSyntax() and then add everything back into the tokenizer, except I add numbers as word characters. The Tokenizer now returns only TT_WORDs, but that's exactly what I want. Now that I have a sequence of tokens, I can do my parsing.

Parsing

The goal of my parser is to fill some fields in the CommandLine object. These fields are used by the Command Processor to find methods, invoke them, and handle the returned object. These fields hold the method name, the target object name (the object that declares the method), the variable name for the returned object, and a collection of arguments.

```
variable = object.method argument,
argument, ...
```

The variable, object, and arguments are all optional. When the StreamTokenizer gives me the first token, I don't know whether I've been given a variable, object, or method. Before I can decide, I'll need to know something about the next token. There isn't any way to look ahead with the StreamTokenizer, so I'll substitute a caching mechanism.

In my parser, the first call to nextToken() must return a word. I can't assign this word to the sMethodName field because I might have a variable assignment. I need to know if the word is followed by "=", a word, or nothing. I have to cache the current token's string value and then call nextToken() again, examining the value. If I find an equals sign, the cached value belonged to a variable. If not, the first word was either "object.method" or "method" and I'll have to put back the token I just took so it will be processed correctly.

Arguments and Types

Immediately following the method is a list of zero or more arguments. Since casting is allowed, they can be quite complex. Arguments can look like 2d, (float)2.33, 3e3, (java.lang.String) "I call it a \"Laser Beam\"", or "This is a test".

> "There has been a fair amount of grumbling about the inclusion of the java.util.Regex package in J2SE 1.4"

For reasons that will be clear later, I need to turn an argument into a Class type and an object representation. When I ran the arguments through the Tokenizer, I got back a string of data, a string type, or both. In the case of the argument string "(float)12", I received "float" and "12". I'll pass these both to my Argument class and let the class handle all the conversions. If I pass a null type to the Argument class, it will try to match the data to a primitive type.

To do that, I match the data given to me from the Tokenizer against the patterns that define the different types of primitive literals in the Java language. For example, "true" and "false" are the only allowed Boolean literals in Java. If I'm asked to construct an argument with a null type and data = "true", I should be able to easily detect that this is a Boolean argument. To examine the data, I'll use regular expressions as provided in JDK 1.4.

There has been a fair amount of grumbling about the inclusion of the java.util.Regex package in J2SE 1.4. Some people claim that since Java Regex packages have been widely available for some time, Sun is just adding unnecessary code bloat to the JDK. Personally, I wouldn't be as likely to use them if they weren't so readily available to me. No matter how you feel, regular expressions are extraordinarily useful things with which every programmer should have more than a passing acquaintance.

There are three basic functions that you use to apply regexes to your strings: Find() is used to match a substring in a string, matches() is used to determine whether an entire string matches the regex, and split() splits a string wherever it finds a match (similar to StringTokenizer). Unless you're receiving your regex strings dynamically, you'll want to precompile your regexes and reuse them, as this will greatly speed up your code. Create a Pattern object by calling its static compile() method. Since it's static, you can declare Pattern fields this way.

```
public Pattern p = Pattern.com-
pile("(true|false)");
```

My regex in this example is quite simple. It must find either "true" or "false". To apply it to a string, I must first create a Matcher. Then I call the matches() method, which only succeeds if it matches the entire string. So both "tru" and "truly" would fail.

```
Matcher m = p.matcher( stringData );
if ( m.matches() )
// it's a boolean literal.
```

I've created regular expressions for most of the primitive literals. I don't need them for string literals or char literals because those are wrapped in "" or ''. If no type has been found for the data, it's checked to see if it's a valid identifier. If so, it will be presumed that the argument passed is a variable that has previously been created in the

Building J2EE and Web Services?

Accelerate development and deployment with AltoWeb

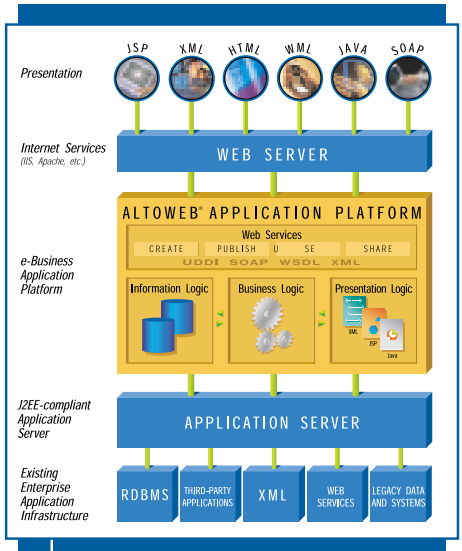
Java™ 2 Platform, Enterprise Edition (J2EE™), XML and Web Services are widely recognized as the foundation for e-business applications. But building, deploying and managing applications on top of application servers is challenging. To succeed, developers need to:

1. Learn and manage a lot of different technologies that are always changing—JSP™, servlets, EJB™, JMS, JDBC™, XML, UDDI, WSDL, and SOAP, to name a few.
2. Choose the right architecture and use the best J2EE practices—because the wrong decisions can lead to performance problems and costly rework later on.
3. Select reusable application components that let them focus on new development.
4. Assemble, test, deploy, monitor, and manage J2EE applications and Web Services.

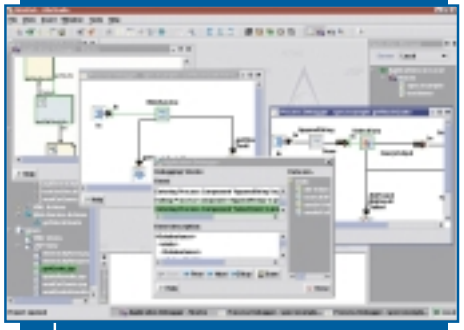
AltoWeb delivers up to 10x faster productivity and time to deployment
The AltoWeb Application Platform lets developers build, deploy, manage, and maintain e-business applications up to 10x faster by replacing the lengthy, custom, and complex coding for J2EE, XML, and Web Services with a packaged solution that shields developers from the low-level complexities involved in building J2EE applications and Web Services. With the AltoWeb Application Platform, developers focus on building 20% of the application specific to their business, and use Java IDEs and Web authoring tools of choice.

Packaged components and reusability out of the box
With AltoWeb, developers get a comprehensive set of pre-packaged EJB components and services that provide the foundation for e-business applications and promote true reusability and maintainability:

- Information logic components enable fast integration to existing enterprise infrastructure elements—including relational databases, file servers, Web servers, and legacy systems.
- Business logic components deliver process control flow and rich functionality.
- Presentation logic lets developers rapidly build rich user interfaces using a variety of open industry standards—including HTML, XML, JSP, WML, and Java.



The AltoWeb Application Platform lets you assemble, deploy, test, monitor, and manage J2EE applications.



Live component assembly shrinks the time spent on assembly, testing, deployment, and maintenance from months to days.

- Built-in support for XML integration and Web Services enables developers to create, publish, monitor, and use Web Services as an integral part of each application.

Accelerated and simplified development, deployment and management
AltoWeb adds the tools development teams need to accelerate and simplify the entire application lifecycle: from development through assembly, testing, deployment, and runtime management.

With AltoWeb, developers move to rapid, visual, live assembly and testing of packaged, reusable J2EE components. They move away from the lengthy, error-prone custom development, integration, component deployment, and low-level testing that consumes most of the time and resources in projects today.

- Application developers who know the business define application logic directly.
- Web page developers focus on the overall look and feel using their Web authoring tools.
- Java developers now focus on developing specialized application logic with their existing Java IDEs.

AltoWeb automates deployment and runtime management by building these facilities into your applications.

Find out more about how AltoWeb has helped its customers become 10x more productive, and deliver applications up to 10x faster. Visit www.altoweb.com to download a complimentary 30-day evaluation copy of our software and get more information.

AltoWeb™

FREE 30-DAY TRIAL!
Download at www.altoweb.com

CommandProcessor. Its type will be discovered just prior to finding the desired method.

To recap, we have now parsed a command line and retrieved a variable name if one was requested, a method name with an optional target object, and a list of arguments. The arguments have one or two pieces of string information. The type can be given by an explicit cast, can be implicit in a literal, or can be implicit from a stored variable. The data is just the string data typed in at the command prompt. The Argument class will have to create an object of the type specified by the argument and "set" it with the given data. It also creates a Class object of the type specified. Why and how Argument does this will be discussed later. First, a little discussion about reflection is in order.

Intro to Reflection

In Java, every instance of an object has an associated "java.lang.Class". Class objects contain the lists of methods, constructors, fields, etc., that belong to objects. These could have been declared in the .java file or inherited. When introspecting an object, you're essentially rummaging through the java.lang.Class information stored in the object. One of the methods built into the Command Processor is "dump". This method will take an object and introspect it and list the constructors, methods, and fields of the given object. It will even try to get the value of the fields returned but, as you'll see, there are few classes that are not fully encapsulated, so you'll rarely see the values in fields – unless you use the trick I show in the Util.toString (Object o) method, as described a bit later.

Unsurprisingly, methods, constructors, and fields are all represented by objects in the java.lang.reflect package and stored as arrays of these objects in the Class object. To get an array of methods for your object, get your object's Class object and call the appropriate get method:

```
Method[] mymethods =
myobj.getClass().getMethods()
```

The same formula works for the fields and constructors. In fact, as I mentioned earlier, I've created a generic toString method utilizing this feature. It's static and I always use it. I simply override my standard toString( ) method with this line:

```
return Util.toString( this );
```

My generic toString method takes the given object and introspects all its fields. Listing 2 shows my toString method being called on the Command Processor. Notice that the RegexMethodFilter also gets introspected. This is because it uses the new toString method. The method will output the field name and the value, if it can get it. Since the field is likely protected or private, toString( ) shouldn't be able to get the value, but that's where the AccessibleObject comes in. Methods, Fields, and Constructors all inherit from AccessibleObject. Simply call setAccessible(true) on the object. This is basically there for things like serialization, but it's worth noting that your private variables are only private if you provide a security manager with your application.

> "No matter how you feel, regular expressions are extraordinarily useful things with which every programmer should have more than a passing acquaintance"

The one thing I should make clear here is the difference between getDeclaredXXXs( ) and getXXXs( ). getDeclaredXXXs gets all the XXXs declared in the class, regardless of access modifiers (public, private, or protected). getXXXs gets only public items as well as all inherited items. We'll go into more detail about making calls on these objects later.

Filtering

Methods contain a plethora of information. There are 12 access modifiers (like public, private, etc.) found in java .lang.reflect.Modifier. Methods always have a name and can also be discriminated by parameter type(s), exception type(s), return type, and declaring class. Filtering on any combination of these properties could be a daunting task but, once again, regular expressions come to the rescue.

To get a list of things to filter, I make a call to one or more of the four "getMethod" members of java.lang .Class. As mentioned before, I have a choice of getting declared or inherited public methods. In addition, I can attempt to find a single method instead of an array of methods. For the Command Processor, this is the most convenient. Any given command line will make clear the name of the method, any arguments, and possibly even the object on which to execute the desired method. The get methods of Class are just the thing for finding a specific method:

```
public Method getMethod(String name,
Class[] parameterTypes)
```

As noted, this searches only public members of the class. Notice that the second argument is an array of class types. This method will return only methods with the exact signature specified by the name and the parameter types. Calls to getMethod are exactly why we needed the Argument class to provide us with a Class object for its type. getMethod must have the exact argument types or it will fail to find a method.

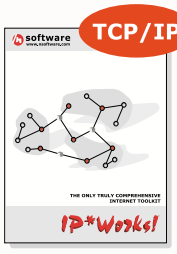
If a method is returned, I still have to check to see if its use is allowed. It's not a good idea to call methods like wait() and run() from the Command Processor, so they should probably be filtered out. The MethodFilter interface abstracts this functionality. The Command Processor instantiates its internal method filter, called RegexMethodFilter, and all objects will use this filter unless another one is provided. The RegexMethodFilter class adds one essential method to the implementation of MethodFilter, addExpression(). This method adds the given expression to an internal list of regular expressions, each of which will be tested against the given method. If a match is found, the method is rejected. This time, we use the find() method of the matcher class because we want to match any substring in the method signature.

The Command Processor, for example, does not want to expose the main( ), run( ), or wait( ) methods, so the internal filter will need to exclude them. The patterns "main\(.* \)", "run\(.* \)", and "wait\(.* \)" will reject main(), run(), and wait(), but not maintain(), runtimeTarget(), or waitlist().

DO YOU HAVE ALL THE PIECES OF YOUR INTERNET PUZZLE?

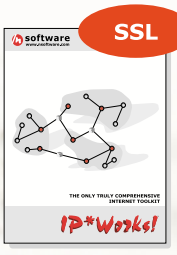
*"...IP*Works! is a network object library that runs everywhere, does all of the things we'd like it to, is easy for all developers to learn, and is well supported by the vendor...."*

InfoWorld Review



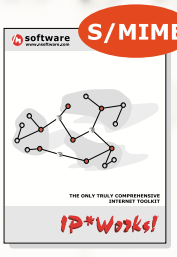
IP*Works!
The Only Truly Comprehensive Internet Toolkit

- More Than 30 Components Cover All Major Internet Protocols
- Follows Exact RFC Specifications
- Market-Tested For Over 6 Years
- In Use By Almost All Fortune 500s
- Royalty-Free Pricing



IP*Works! SSL
Secure Communication Channels Through SSL Security and Digital Certificates

- SSL-Enabled Versions of the IP*Works! Components, plus a Digital Certificate Management Component
- Secure HTTP, Secure SOAP, Secure Mail, Secure Servers, etc..
- Royalty-Free Pricing



IP*Works! S/MIME
Email and Document Security, Integrity, and Non-Repudiation Using Digital Signatures

- S/MIME Enabled Versions of the IP*Works! Email Components, plus a Generic S/MIME V2 Component
- Encrypt, Decrypt, Digitally-Sign, and Verify Documents, Emails, News Articles, etc..
- Royalty-Free Pricing

software
www.nsoftware.com

THE #1 PROVIDER OF PROFESSIONAL INTERNET COMPONENTS

Running the Command Processor

Requirement 5 calls for internal commands to provide the most basic functionality, like exit and list. Since I know that the Command Processor will introspect an object looking for methods to call, I'm going to get slick and have the Command Processor introspect itself for the internal methods. This way, if I choose to add a new method to the list of internal commands, making it available is as simple as ensuring that it doesn't get filtered out.

Parsing, etc.

Once the Command Processor has loaded the internal and external filters and objects, the run command enters a loop, grabbing a string of input from stdin, parsing it into a command and parameters, and finding and invoking the underlying method. As discussed earlier, the CommandLine class takes the input from the user and breaks it into a method name and a collection of arguments. It also provides access to the arguments as an Object[] and as a Class[]. As we'll see, this is not exactly trivial. It's important, however, because you invoke methods with an object array but find them with a class array.

There is quite a bit of chicanery involved with handling parsed parameters and it's all because of primitive types. The reason, as I mentioned earlier, is that the "getMethod" members of Class all expect a Class[] to describe the arguments of the method you want. The "invoke" member of Method requires an Object[] with correct types and data. Making classes and objects for primitive types is a bit tricky.

Let's examine the process for parsing "(float)21". We see that we have a parameter with the type "float" whose data is "21". I quote them here to remind you that they're still strings. float is a primitive type and there's no facility to turn a primitive into a Class object. In a perfect world, you'd be able to dynamically create your primitive the same way you do any other class: Class myClass = Class.forName("java.lang.StringBuffer"; . Once you have a class, it's trivial to create an object if it has no argument constructor: Object myObj = myClass .newInstance();. Unfortunately, this is not allowed for primitives. There are static class objects available for the primitive types and they must be used here. They are members of the classes that wrap primitives. For int, there's the java.lang.Integer class; use its TYPE field as shown in Listing 3.

Now I'm faced with a nice long list of if/else statements. I actually used a static hashtable instead, which may be a bit faster and is much more flexible. At this point, I should have enough information in CommandLine to find the named method from a class. In the Command Processor "run" method, I try to get the method from the Command Processor, from the named object (if specified), or from the "target" object defined when the Command Processor was constructed.

If the target wasn't named (i.e., myobj.myMethod), the processor goes first. This way, the processee can't accidentally override the exit command and get you stuck in a loop (experience teaches me yet another hard lesson). Once I have a method, it's a simple matter of invoking the method and providing feedback to the user.

Invoking

"Ay, there's the rub" – Wm. Shakespeare

We've already had a quick look at Method's "invoke" method and the complete signature is shown in Listing 4. As you can see invoke takes two parameters. The first, an Object, is an object of the same type as the method's declaring object or an object of the same type as a subclass of the method's declaring object. This means you don't need the same object that gave you the method – any object of the same class or subclass will do. In our case, we're finding the method from the object we'll invoke it on, so we'll have no problem meeting that requirement. Some test code is provided with the full source that shows a simple walkthrough of what you can and can't do with invocation. More complex scenarios might have you invoking the same method on a variety of different objects that implement the same interface. Not having to find the method from each individual object would surely be convenient and faster.

The second parameter to the invoke method is an array of objects, one for each parameter and each representing an argument. The catch is you can't magically create an object for a primitive type. I really don't even have a primitive type. I have a string representing a primitive type and a Class object. As we've already noticed, I can't create an object of a primitive type via the Class object. I have to determine the type and create an instance of the appropriate wrapper class. For the int type this is java.lang.Integer. Therefore, it looks like I'll have another battery of if/else statements.

```
if ("int".equals(sType))
 Object o = new Integer(sData);
```

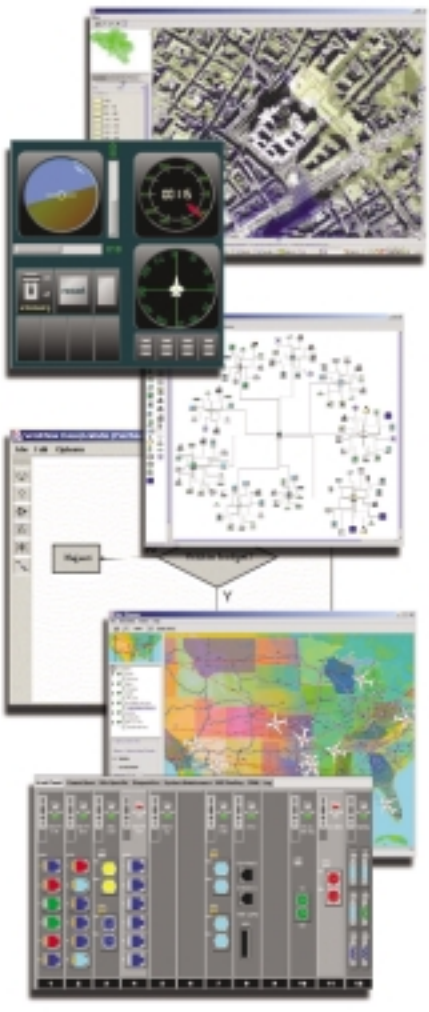
Once again, rather than a long if/else block, I use a static hashtable. This time it's a bit more complicated, but I can take advantage of the fact that all primitive wrapper classes have string constructors (except char, which is handled as a special case). Notice that I dynamically search for the string constructor to invoke. The advantage of that strategy is that any object with a string constructor can be instantiated in the same step. For example, one of the API class's methods, test(java.lang.StringBuffer), works automatically with this setup.

As you can see, the method I found was located with a class object of type int (from the static Integer.TYPE), not with a Class object of type Integer. They are different. They have to be able to discriminate between foo(int I) and foo(Integer I). However, when I invoke the method, I use objects of the wrapper types. The JVM will handle the conversion for me, but it's important not to confuse the methods. In essence, you get around strong typing here so be sure you're calling the right method. Again, the full source provides sample code that shows argument conversion at work.

One final note, the invoke method declares three exceptions. Of course, when you dynamically call a method, you can't specify any exceptions thrown by the method, because you

"There is quite a bit of chicanery involved with handling parsed parameters and it's all because of primitive types"

Accelerate Your Java 2D™ Graphics Development



Powerful Java™-based visualization components for mission and business critical management applications

...without the runtimes.

JLOOX Visualization Suite* is a collection of high-performance Java 2D and Swing-based graphical components for building virtually all kinds of management systems and visual applications:

- Transportation systems management
- Network and element management systems
- Workflow, business process and supply chain management
- Custom drag and drop style interactive editors
- Prototyping, testing, and operator training

Deploy in Java applications, applets or servlets.

Join the hundreds of companies using **JLOOX Visualization Suite** worldwide by visiting our website today to request a free 30-day product evaluation.

1.800.361.6424 **www.JLOOX.com**

***JLOOX Visualization Suite originally developed by LOOX Software (recently acquired by eNGENUITY Technologies).**

eNGENUITY Technologies

Visual. Software. Solutions.

don't know what method you'll be calling until runtime. InvocationTargetException wraps the exceptions thrown inside the method. If you want to know which exceptions the method threw, you have to call the new method in Throwable, "getCause". This is part of the enhanced "Chained Exception Facility" in JDK 1.4, which is a standardization of chaining in the Throwable class.

Kicking My Own Tires

Listing 5 provides sample output from the Command Processor with the API class provided. List is an internal command that lists the internal commands available and those exposed by the internal target object. It also calls listvars, which will list any variables assigned. In the API class, test : java.lang.StringBuffer merely takes the input and calls reverse on it, printing them both. There are two Test methods (notice the capital "T"). One takes an int and the other takes an Integer. The int version adds three to the given number and the Integer version adds four. Both return ints. Notice that the second call, which uses the variable assigned from the first call, ends up calling the Integer version. This is because invoke always returns an object. There is no way to determine the primitive type of the returned value from invoke without checking the Method object. I haven't bothered to catch that yet, but it should be simple enough. The mischievous among you might want to try casting the variable directly into an int, like this: Test (int)myvar. That's going to throw a NumberFormatException because the constructor to int will not be able to turn "myvar" into a number.

Since I don't want the Argument class to know anything about the variables or even the Command Processor, I can't resolve this in the Argument class. The smart thing to do would be to turn the returned object into an Argument and store arguments in the variable map rather than objects. This is left as an exercise for the reader (I've always wanted to say that).

Conclusion

As you browse through the source of this project, you'll likely note that most of my classes have main functions with a variety of test scenarios in them. Most of the tests are commented out, with only the latest test left standing. This is because I'm a firm believer in adding a feature and then testing it right away. Perhaps, now that I have completed this tool, I'll find myself writing fewer main functions. What I'm really looking forward to is writing a whole lot less System.out.println( ) statements. Now, if I want to know the state of an object, I'll ask it directly.

This code is easily adaptable to an internal console similar to those found in most PC games these days. Anyone who has played a PC game in the last five years has likely seen the drop-down command prompts that are becoming ubiquitous. The console, while not new by any stretch of the imagination, provides a very useful tool for developers and power users. Normally, a console would allow only the getting and setting of parameters and the reloading of configuration files. However, even viewing and setting properties in the running system can be extraordinarily useful.

Special Thanks

I would like to thank the following people for their contributions to this article: Brett Andrews and David Pidcock for the original concept and reflection ideas, respectively. Mark Eames, David Colon, and Jerome Liang for the technical review and numerous English corrections.

Author Bio

Richard Ross is an engineering manager for Raining Data, Inc. He has been an engineer for 16 years, with experience ranging from custom ASICs to enterprise Java development.

rross@stupendousman.com

SpringLayout: A Powerful & Extensible Layout Manager

Simplify the visualization and implementation of GUIs

written by Joe Winchester and Philip Milne

The task of a layout manager is to position and size each component based on the size of its container. Each component has a preferred size that can be used to determine the real estate it wishes to occupy, as well as a minimum and maximum size. The preferred size is especially useful for components that contain user-visible strings whose size can change between development and deployment due to different fonts or different locale values, as the GUI is dynamically translated through resource bundles.

Layout managers are powerful classes for dynamically recalculating the size and position of the GUI components at runtime; however, they often lead to a less than optimal design-time experience. With WYSIWYG development tools, it frequently happens that developers position the mouse where they wish to drop a new component, only to see the entire GUI reorganize and the component placed in another location. This is due to the indirection involved in translating the developer's gesture to the correct constraint for the layout manager.

JDK 1.4 introduces a new layout manager – javax.swing.SpringLayout. The motivation behind building it was to create a layout manager that made it easier to design GUIs, was powerful enough to emulate the behavior of all the existing JRE layout managers, and be easily extensible where a new rule was required.

The most pleasant user experience with a design tool occurs when you set the layout manager to null. Then you can place your component exactly where you want it using the mouse and set its size using resize handles. The problem with not having a layout manager is that the explicit bounds of the component are fixed constants set at design time. With SpringLayout you can specify each component's *x*, *y*, width, and height, but instead of using fixed coordinates, the constraints are springs. These springs can be derived from the edges of existing components, and are able to flex under tension and compression when the window size is changed.

To illustrate SpringLayout this article demonstrates how to use springs to anchor edges of components and align widths, and how to create complex springs.

Background

To specify how each component should be positioned and sized by the layout manager you must use a constraint object, which is an instance of the inner class javax.swing.SpringLayout.Constraints. A SpringLayout.Constraints object contains six springs – for the component's width and height and each of its four edges.

A spring is analogous to a physical spring that connects the edge of the component to another point on the GUI. Each edge is controlled by a single spring, so springs are best thought of as directional. For example, a spring that fixes the left edge of a component to the right edge of another doesn't operate in reverse. When the first component's right edge moves, this affects the second component's left edge, but not vice versa.

Springs can be fixed struts that have a constant value, or else can be flexed under tension and compression. To determine how elastic a spring is, it has a preferred, minimum, and maximum value. If the minimum value is the same as the preferred value, the spring is rigid under compression, and, likewise, if the maximum value is the same as its preferred value the spring can't be expanded. If no spring is specified for a component's width or height, the default spring created defers its preferred minimum and maximum values to the component. In addition to using individual springs to anchor and size components, springs can be combined to create compound springs whose elasticity is the sum, maximum or minimum, of its arguments.

When designing a GUI using SpringLayout, draw a diagram of the components in the positions you want them to be in, and then mentally determine what will keep them in those positions. Usually this will be a fixed length spring between the component's left and top edge relative to its nearest neighbor's edges. This is very useful for components that contain user-visible strings, such as buttons and labels where the exact size isn't known until runtime, because the font used or the

sitraka

JProbe®

Jolt your application's performance.

software development 2002 product excellence award JOLT

Award-winning Sitraka JProbe provides deep J2EE diagnosis tools that give you a complete picture of your Java application, right down to the offending line of code. JProbe includes everything you need for fast, accurate analysis of your servlet, JSP and EJB code.

New integration wizards, improved investigation features and sophisticated data filtration options now make JProbe even better. See for yourself why **JProbe won a 2002 Jolt award.**

- JProbe Suite
- JProbe Profiler
- JProbe Memory Debugger
- JProbe Threadalyzer

- JProbe Training
- Unparalled OS, JDK, application server and IDE support

For more info and a free eval, visit:

www.sitraka.com/jprobe/jdj

sitraka

the Java™ advantage

FIGURE 1: The OK button with fixed springs for x and y

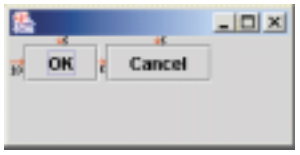


FIGURE 2: Cancel button 5 away from the OK button and the top of the contentPane

FIGURE 3: The window is sized based on the springs between it and the Cancel button

FIGURE 4: The OK and Cancel buttons have their widths aligned

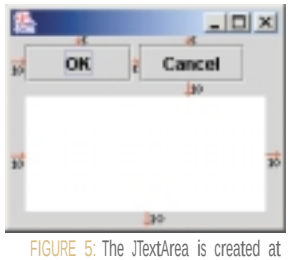


FIGURE 5: The JTextArea is created at its preferred size

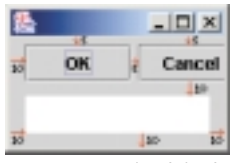


FIGURE 6: The window is reduced and the springs remain rigid

FIGURE 7: The flexible springs compress as the window size is reduced

actual string, if it has been externalized in a locale-specific bundle, can vary.

To show how to use springs, this article builds a simple GUI that illustrates some of the common types. The screen images have red lines drawn on them to help you visualize each spring. This approach, drawing springs between components, is an intuitive way to design a GUI where you begin by drawing the components at their absolute positions and sizes and then add lines to represent the springs. A GUI builder could employ this method where you drop a control at the position you want it placed, and springs are automatically created between it and its nearest neighbors.

Constant Springs

The simplest spring is a fixed value that positions an edge a constant distance from its parent container. Figure 1 shows a button that has two constant springs used to position it at 5,10 inside a frame's contentPane. The SpringLayout class automatically generates a Constraints object for every component that's added to its container, so in order to change any of the springs you can query the Constraints object and set new values. To create a constant fixed spring, use the static method Spring.constant(int).

```
JButton okButton = new JButton("OK");
contentPane.add(okButton);
SpringLayout.Constraints okCst =
 layout.getConstraints(okButton);
okCst.setX(Spring.constant(10));
okCst.setY(Spring.constant(5));
```

Springs Between Edges

In Figure 1 the springs for the OK button were fixed lengths of 5 and 10. Even though no springs were specified for the width and height of the button, some were created automatically based on the preferred, minimum, and maximum size. These springs can be queried and used in the constraints of other components so you can position them relative to each other. For example, a Cancel button can be added to the content pane, whose *x* spring is the right-edge spring of the OK button. To query the spring for an edge, the method public Spring getConstraint(String edgeName) can be used that takes arguments of "North", "East", "South", and "West" for each edge. To position the Cancel button so its left edge is a fixed distance away from the right edge of the OK button, a compound spring is used. The method Spring.sum(Spring spring1,Spring spring) creates a compound spring whose value is the sum of two arguments. By combining the east edge of the OK button with a constant spring of 5, a spring is created that will always be five larger than the right edge of the OK button. This spring is then used as the *x* spring of the Cancel button.

```
JButton cButton = new JButton("Cancel");
contentPane.add(cButton);
SpringLayout.Constraints cCst =
layout.getConstraints(cButton);
cCst.setY(Spring.constant(5));
cCst.setX(Spring.sum(
Spring.constant(5),
okCst.getConstraint("East")));
```

The result of this is shown in Figure 2 where the Cancel button is positioned to the right of the OK button with a space of 5 pixels in between.

Sizing the Window with Springs

In addition to allowing you to specify constraints for a container's components, you can specify a constraint for the container that allows you to control its edges. The content pane's right edge can be set with a compound spring as 20 away from the right edge of the cancel button. To do this, retrieve the spring for the layout manager's container, and then set its east spring.

```
SpringLayout.Constraints pnlCst =
layout.getConstraints(contentPane);
pnlCst.setConstraint("East",Spring.sum(
Spring.constant(20),
cCst.getConstraint("East")));
```

For the content pane's bottom edge a spring can be put between it and the bottom of the buttons. This spring can make the bottom of the contentPane be a distance of 15 away from the buttons, and because the OK and Cancel buttons are the same height and have the same *y* spring, the content pane's bottom edge can spring from either one. A compound sum spring is created that adds the south spring of the cancel button's constraint to a fixed spring of 15.

```
pnlCst.setConstraint("South",Spring.sum(
Spring.constant(15),
cancelCst.getConstraint("South")));
```

When there are springs controlling the edges of the contentPane, the window can be packed to give it an initial size (see Figure 3). In the previous examples where there were no springs for the edges of the contentPane, the window was given a fixed size of 200 by 200.

The content pane's edges can have a spring between them and a specific control, such as the Cancel button. However, if there is no single control that determines the edges of the content pane, a compound spring can be created. This compound spring needs to determine which is greater, the south spring of the OK button or the Cancel button. The method Spring.max(Spring spring1, Spring spring2) is used to create a single spring whose value is the greater of the two arguments.

```
pnlCst.setConstraint("South",
Spring.sum(Spring.constant(15),
Spring.max(
cancelCst.getConstraint("South")),
okCst.getConstraint("South")));
```

Springs for Widths and Heights

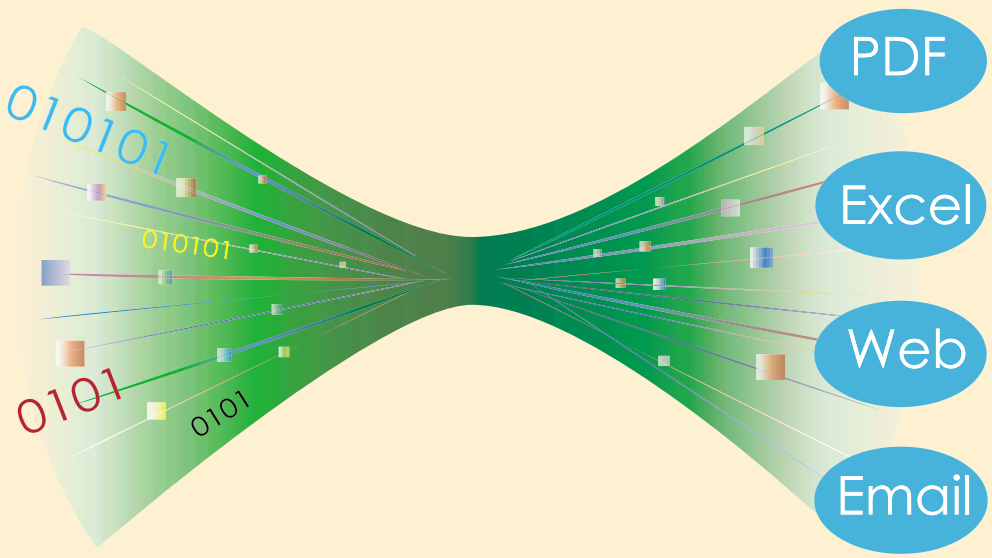
You can also use springs to set a component's size, as well as its position. For example, instead of the OK and Cancel but-

My Enterprise Reporting Solution Must:

(1) be built on open-standards

(2) support end-user ad-hoc queries on the web

(3) have reusable logic/ components



STYLE REPORT

InetSoft provides open standards based, innovative component technologies. Style Report is a 100% Java, integration oriented retrieval solution that aggregates data from any source into intelligent, actionable, decision making information. Flexible delivery channels, such as Web, email, PDF, and Excel ensure that information is available where and when it's needed. Zero client Ad-Hoc web reporting is provided inside a comprehensive framework that maximizes end user's self-service productivity. Style Report instantly becomes a seamless part of any infrastructure.

STYLE REPORT

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

www.inetsoft.com

InetSoft

ton each being their preferred size, you want them to be the same size. The size used will need to be the larger of each button's preferred size so that neither label is clipped and a compound spring can be created based on the maximum of the two widths.

```
Spring widthSpring = Spring.max(
okCst.getWidth(),
cancelCst.getWidth());
okCst.setWidth(widthSpring);
cancelCst.setWidth(widthSpring);
```

Because the code changes the spring that specifies the width of the OK button, and this spring is also used to calculate the distance between the OK and Cancel buttons, the constraint between the two buttons must be defined after setting their width springs. This ensures that the *x* spring of the cancel constraint is not based on its original width spring, but is derived from the max spring instead. The order of setting springs that form input to compound springs is important, and the following code must be done after the OK button's width spring is set.

```
cancelCst.setX(Spring.sum(
Spring.constant(5),
okCst.getConstraint("East")));
```

Likewise the spring that's created between the content pane's right edge and the right edge of the Cancel button must be defined after the width spring is established. Generally, it's a good idea to specify any explicit width or height springs before defining the positional springs between components to ensure that no queried width or height springs are later replaced. Because the word "Cancel" is longer than "OK", the OK button's width is increased to match that of the Cancel button. However, if in a different language the lengths were reversed, the two buttons would always remain the same width. Figure 4 shows that the OK and Cancel buttons are the same size, and that the overall window size still remains 20 and 15 away from the Cancel button.

The ability to compute a component's width and height using a compound spring gives you a level of control not previously possible with other layout managers. In GridBagLayout you can align widths of components by placing them in the same column, but with SpringLayout you can align widths where the controls are side by side as shown in Figure 4. Sometimes the requirement for laying out buttons may be a rule such as "Make the buttons have the same width but make this be no smaller than 60." This can be created by having a compound spring such as:

```
Spring widthSpring = Spring.max(
Spring.constant(60),
 Spring.max(
 okCst.getWidth(),
 cancelCst.getWidth()));
```

Components Resizing with the Container

For some components, such as JTextArea, the desired layout is to make as much use of the available space as possible. To do this springs can be made between the east and south edge of the text area and its container. Listing 1 creates a JTextArea and adds constant springs between its east edge and the container, and its south edge and the container. The lines of code setting the springs between the Cancel button can be deleted as the contentPane's constraints are attached to the text area; although, if left there the panel will use the newest springs since only one spring can be held for each constraint in the SpringLayout.Constraint class, and setting a new one replaces any existing spring.

The panel opens with a default size based on the preferred size of the JTextArea (see Figure 5).

When the panel is resized, the JTextArea will be resized with it, expanding to make use of extra width or height. This occurs with the JTextArea and not with the OK and Cancel buttons due to the default width and height springs that were initially created for each component by the layout manager. A JButton's preferred, minimum, and maximum sizes are all the same value, so there is no "give" in an arrangement of buttons connected together with "struts." By contrast, a JTextArea's minimum and maximum and size are not the same, so the default width and height springs can expand and compress. After the panel changes size, the layout manager looks at the flexibility of all its springs and, based on their elasticity, calculates their values.

The JTextArea becomes larger when the window size is expanded, and smaller when the size is reduced. The fixed-edge springs remain at their value of 10, while the JTextArea shrinks to occupy the remaining space.

With a reduced window size the springs between the edges of the JTextArea and its neighboring edges remain 10 pixels wide, and the JTextArea reduces in size because its minimum size that provides the input to its width and height springs is 0 (see Figure 6). However, rather than have the springs between the JTextArea and its neighbors rigid, they can be made flexible so they'll compress when the window size is reduced. This way, when the window is made smaller, more of the actual space can be given to the JTextArea.

> "The rules by which the layout manager determines how much to compress each of the springs are based on how much it resists being compressed"

In addition to defining a single value for a constant spring, you can also set a minimum and maximum size, e.g., Spring.constant(0,10,20). If the minimum and maximum values are the same as the preferred value, which occurs when the single argument constructor is used, the spring becomes an inflexible strut. However, with a minimum and maximum size, when the window size is changed, the layout manager takes these values into account to determine the new actual values of each spring. To illustrate this all the springs used in the example between the JTextArea and its neighbors were set to be constant with a minimum of 0, a maximum of 20, and a preferred size of 10.

```
txtCst.setY(Spring.sum(
Spring.constant(0,10,20),Spring.max(
```

```
cancelCst.getConstraint("South"),
okCst.getConstraint("South"))));
pnlCst.setConstraint("East",
Spring.sum(Spring.constant(0,10,20),
txtCst.getConstraint("East")));
pnlCst.setConstraint("South",Spring.sum(
Spring.constant(0,10,20),
txtCst.getConstraint("South")));
```

What occurs now is that when the window is reduced in size, the springs can be compressed. The layout manager looks at the compressibility of each spring, including the implicit spring that the JTextArea has for its width and height, to determine the size and position of the controls. When the window is made smaller, the springs compress and the JTextArea occupies more of the available real estate. When the window size was reduced to a height of 77, the JTextArea, with rigid springs around it, was made 26 high, enough to show two lines of text. However, the flexible springs mean that they can take the strain and allow the JTextArea to have a larger size (see Figure 7). The top and bottom Springs are compressed to a value of 4, and the JTextArea has 12 extra pixels available for its height, making it 28 high so it can display an extra row of text. If flexible springs were also used for the top of the two buttons, the window could be further optimized so that yet more of the more available space was allocated to the JTextArea under compression.

The rules by which the layout manager determines how much to compress each of the springs are based on how much it resists being compressed. When a spring is compressed, imagine that it provides an opposite force that increases as it reaches its minimum value. If the minimum and preferred values are the same, the force is an irresistible one and the spring is rigid.

The springs between the edges have a minimum value of 0 and a preferred of 10, and the JTextArea's height spring is based on its minimumSize's height of 0 and preferredSize's height of 5 rows of characters at the current font. Both of these springs are compressed as the window size decreases. When the window is increased, the edge springs have a maximum size of 20 and a preferred size of 10, however, the maximumSize of the JTextArea is based on the look and feel but is likely to be Integer.MAX_VALUE. Because the resistance of a spring increases as its value approaches its maximum value, for the JTextArea's width or height to become rigid the window would have to become enormous. Use of compressible springs between components allows you to create a GUI that reallocates its available spaces across its components as it changes size, so the user has more available working component area and the window remains effective over a longer range of overall sizes.

Convenience Methods for Attaching Springs Between Edges

To attach the Cancel button to the OK button and make them 5 apart:

```
cancelCst.setX(Spring.sum(
 Spring.constant(5),
 okCst.getConstraint("East")));
```

This scenario, where you attach the edge of one component a fixed distance from another, occurs frequently when creating GUIs with SpringLayout, so there's a helper method

```
SpringLayout.putConstraint(String targetEdge, Component
targetComponent, int distance, String sourceEdge,
Component sourceComponent);
layoutManager.putConstraint("East",
 cancelButton , 5, "East", okButton);
```

Creating Springs in the Correct Order

Because of the cyclic nature of springs, i.e., they are often calculated based on existing springs that are combined to create new compound springs, the order in which the constraints are set is important. For example, if you query the west or width spring of a component, should you then set its east spring, afterward this could affect either of the previously queried springs, making their values stale. Therefore, it's important to establish and query strings in the correct order. The analogy is that of a spreadsheet whose cells' values are based on calculations that include other cells' values as input.

An application window is made up of a set of components, and SpringLayout lets you add a set of springs into this picture. The springs define the positions and sizes of components, and can be anchored to other components' edges, or can be derived from calculations using existing springs.

We hope that you'll have some fun with SpringLayout and that it helps solve some of those difficult layout problems where things don't work out quite the way you want. In our minds at least, the holy grail of layout experiences would be a GUI builder that would allow the user to add constraints between components visually – rather than having to write code. This is quite a difficult problem; if you're working on a GUI builder or have ideas about the best way to do this – drop us a line!

Resources

- *Online tutorial of SpringLayout:* http://java.sun.com/docs/books/tutorial/uiswing/layout/spring.html
- *JDK 1.4 SDK download:* http://java.sun.com/j2se/1.4/download.html

winchest@uk.ibm.com

philip.milne@drkw.com

Author Bios

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

Philip Milne is a software developer who worked at Sun as part of the Swing development team, and now works as a consultant in London.

Listing 1

```
JTextArea txtArea = new JTextArea(5,15);
contentPane.add(txtArea);
SpringLayout.Constraints txtCst =
 layout.getConstraints(txtArea);
txtCst.setX(Spring.constant(10));

txtCst.setY(Spring.sum(
Spring.constant(10),
Spring.max(
cancelCst.getConstraint("South"),
okCst.getConstraint("South"))));

pnlCst.setConstraint("East",Spring.sum(
Spring.constant(10),
txtCst.getConstraint("East")));

pnlCst.setConstraint("South",Spring.sum(
Spring.constant(10),
txtCst.getConstraint("South")));
```

community.isavvix.com

Your eLearning Playground...

... learn, play, work, and grow with 68,000+ Java developers!

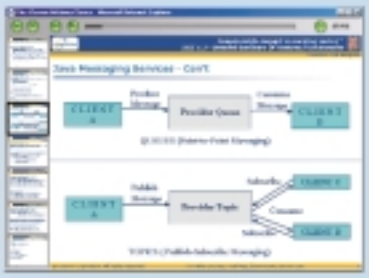
WINNER of "Outstanding Contribution to the Growth of the Java Community (Group/Organization)", JavaOne 2001

NEW AND VERY COOL!

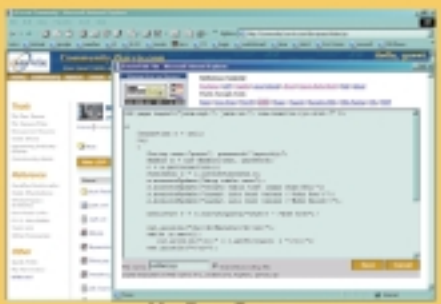
1-Hour, High-Impact online courses.
Pre-Recorded, Instructor-Led.

Introducing 6 new courses

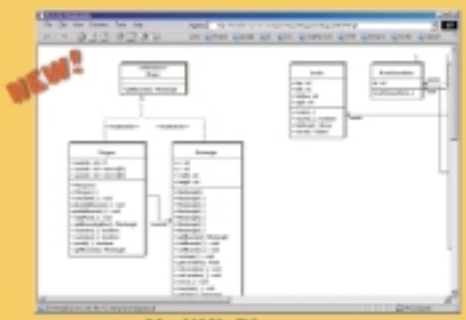
- Java Best Practices
- Java Performance Tuning
- J2EE Architectures
- The Struts Framework
- Getting Started with XSLT
- J2EE 1.3 - A Detailed Summary



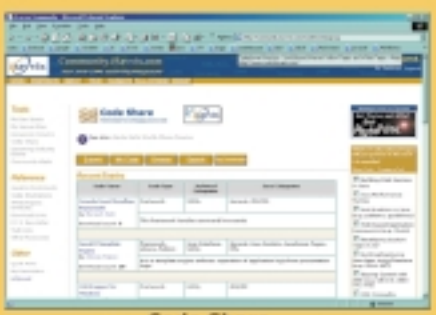
Plus FREE Tools and Resources designed for YOU!



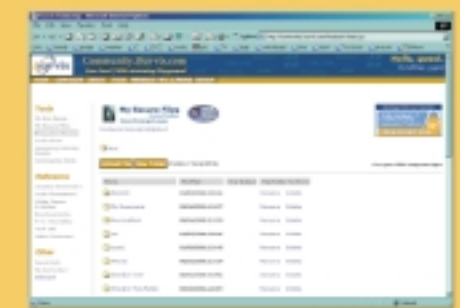
My Dev Space
Free JSP Hosting with JDBC, XML/XSL, J2SE 1.4, tag libs...



My UML Diagrams
UML diagrams on the fly; software from Softera.com



Code Share
Central place for exchanging source code



My Secure Files
Encrypted, compressed file storage/access

MORE:

Forums
Industry Events
Code Illustrations
White Papers
Downloads
Links
Tech 101
Polls
Reminders ...

About iSavvix
iSavvix is an information-technology consulting and services company that provides custom solutions based on Enterprise Java™, XML and related technologies. Since its inception in 1996, iSavvix has built robust, secure, user-friendly, and cross-platform enterprise applications for many Fortune 1000 and other leading companies.

iSavvix
Delivering Java Technology Solutions for your eBusiness

iSavvix Corp. Herndon, VA. +1-888-iSavvix, info@isavvix.com

JASON R. BRIGGS J2ME EDITOR

The Latest in Mobility: Something for Everyone

I recently noticed that QUALCOMM has licensed the ARM1136J-S microprocessor core. The interesting thing to note about this announcement is one of the letters in that microprocessor version: the "J". ARM's 1136J-S is a Jazelle-enabled chip, meaning it's optimized for the Java instruction set; QUALCOMM has effectively licensed a processor that will run Java applications more efficiently.

This is not as strange as it sounds: BREW already seems to support Java applications. However, what is not clear is how much they will charge for certifying those applications, bearing in mind their standard certification charges. I'm hoping QUALCOMM takes the high road and doesn't charge exorbitant fees for MIDP applications running on their BREW phones. I see the necessity for testing C/C++ code but, as I mentioned last month in my editorial "Has Hell Frozen Over?" (Vol. 7, issue 11), there's no good reason why Java should require the same level of testing. Perhaps you've had experience with BREW; let me know your thoughts.

Nokia has an interesting lineup of new phones (I've already rewritten my letter to Santa accordingly), and all of them appear to be Java-enabled, with the exception (I believe) of the entry-level model. Personally, I think it's unfortunate that Nokia chose to omit MIDP from their most basic phone – it seems to me they're missing a large potential market for Java applications. The teenage/underage market might, in general, want to have the most fashionable (and, consequently, expensive) phones available, but there's a (probably quite high) percentage that won't be able to afford anything but the cheapest. And, as I think I've said before, teenagers are initially the one group likely to get the most use out of downloadable Java applications – especially on the entertainment side of things. Think Pokémon/Digimon, or another of those turn-based monster-training/card swapping/etc. type games, played out over a network among thousands of players…

Speaking of entertainment, another Nokia product announcement caught me by surprise. The N-Gage is a "mobile game deck" that looks rather like a Gameboy Advance but, according to some reports I've read, is not aimed at competing with Nintendo's device. I realize every analyst worth his or her salt is predicting that money will be falling from trees in the wireless games market – we'll all be stuffing it into sacks, using it in our mattresses, and, my own personal favorite, swimming through piles of cash like Scrooge McDuck. Therefore, we're told, the market is potentially large enough to support two, or even more, hand-held games machines.

Inevitably, however, comparisons between the two devices will be made, and you have to wonder how well the N-Gage is going to hold up against a far more established GBA. Of course, one factor in Nokia's favor is that Sega will apparently be developing games for the N-Gage – and it will be interesting to see whether a majority of those games are native to the Series 60 Operating System, or if they're Java-based. If you're interested in learning more, see the excellent Infosync site for more details and a picture of the N-Gage, www.infosync.no/news/2002/n/2548.html.

• • •

In this month's ***JDJ***, Karl McCabe, from Rococo Software, investigates some of the issues surrounding ad hoc networks, while Jeremy Wakefield, Keith Braithwaite, and Tony Robinson look at some of the real-world problems facing MIDP developers.

• • •

One final note: if you're just getting started with J2ME, and bewildered with the array of options that sit beneath the Mobile Edition "umbrella," check out the "Survey of Java Today" at the Wireless Java Web site (http://wireless.java.sun.com/getstart/articles/survey/) – an excellent summary of the various technologies and how they're put together.

jasonbriggs@sys-con.com

AUTHOR BIO

As well as being the J2ME editor for Java Developer's Journal, Jason R. Briggs is a Java programmer and development manager for a wireless technology company, based in Auckland, New Zealand.

J2ME INDEX

QUALCOMM®

Name:	Hartmut Neven
Company:	Eyematic®
Most recent titles:	Shout Messenger™ & MyGenie™
Dream:	To make wireless communication as visual as it is verbal.

As CTO of Eyematic, Hartmut Neven is responsible for creating the most viable and cutting-edge wireless applications imaginable. Two prime examples are the new BREW™ applications Shout Messenger – the first U.S. multimedia messaging app to wirelessly deliver animated content – and MyGenie Daily Horoscope. "BREW is extremely convenient for the developer because it's a one-stop solution," Neven said. "After an app is tested, it is immediately made available to all other BREW carriers, and consumers can literally begin downloading within hours." And Neven isn't alone. Commercial services are launched and BREW applications such as games, email, news, weather, stock trades, position location and ringers are now heading to market – a market of millions upon millions. If you aren't developing for BREW, you aren't developing to your potential. To get started, go to www.qualcomm.com/brew.

brew™
Customize. Personalize. Realize.™

Run Once, Right Anywhere?

Minimize portability issues

WRITTEN BY
JEREMY WAKEFIELD, KEITH BRAITHWAITE, & TONY ROBINSON

For Java technology to fulfill its promise, it should be possible to test a Java application on just one device and, if it works properly, assume that it will run well on any other.

AUTHOR BIOS

Jeremy Wakefield is a design consultant specializing in UI design and audio software. He was head of product software for Psion and later a design manager for Symbian.

Keith Braithwaite is technical lead for mobile software projects at Penrillian. He focuses on Java and C++ development with an emphasis on design, quality, and process. He is an active member of the XP community.

Tony Robinson is project manager at Penrillian with over 5 years of experience delivering tools, components, and frameworks for mobile platforms.

J2ME, in its most familiar guise of MIDP, comes closer to this ideal than either the Standard or Enterprise edition because there's only one version of the specification. Although there are different implementations, as far as the Java application is concerned it isn't able to tell which platform it's running on. There is a fly in the ointment, however. While MIDP offers a highly consistent runtime environment and so consistency of functionality, user interfaces vary wildly.

Meanwhile, on Planet Earth . . .

Real-world marketing issues mean that manufacturers produce phones in a range of shapes and sizes, with capabilities to suit every pocket. Can developers be blamed for wanting to take full advantage of the facilities of each individual device? Instead of forcing owners of high-end devices to make do with the minimal interface the J2ME standard requires, developers would, for instance, like to make full use of the large, deep-color screens that the latest devices sport.

Yet network operators (who aim to be the premier suppliers of mobile applications and services) don't just want portability; they demand it. They will, for many consumers, also be the first point of contact when technical problems arise. If a MIDlet behaves very differently across various devices, the technical support costs for the operator could spiral out of control, and these costs subsequently will have to be passed on to the consumer, the developer, or possibly both.

Obviously then, it's in the software developer's interest to minimize portability issues as much as possible.

Going Green . . .

The assumptions that MIDlet authors can make about the device running their applications are deliberately limited, and they have no control over application appearance. They do know, however, that their applications might be deployed on a device with only a numeric keyboard, a 96 x 54 pixel, and a 1-bit deep screen; or they might be deployed on a device with a _ VGA display, qwerty keyboard, and touch screen. Or anything in between. This has led to several different application styles.

The first style collects raw input events and uses Graphics, with its 2D primitives, directly for the UI. This style is typical of the games that currently form the bulk of available MIDlets and is aimed at smartphones. With these applications, assumptions about screen size, aspect ratio, and color depth are crucial factors when producing a portable app. These applications can be memory- and processor-hungry.

The second style of MIDP application uses displayables and commands to build simple form-based UIs, most similar to those on old green-screen mainframe apps. The commands are presented to the user in the same way that commands are usually displayed on the device. In principle, this offers the highest level of portability, since the items in the UI, and the commands that affect them, are presented to the user using implementations native to each device. As we shall see, there are problems with this approach.

A third style combines the first two approaches using a library such as kAWT to provide something closer to a traditional UI, with stronger layout control and richer UI components. This allows a great deal of portability at the expense of increased memory consumption and extra processor load. Application designers still need to make some assumptions regarding screen size.

All for One, and One for All . . .

To be allowed to bear the distinctive Java logo and use the phrase "Java powered," a device has to pass the tests in the Technology Compatibility Kit (TCK) from Sun Microsystems. There's just one TCK for all devices. This guarantees conformity to the Java standard.

However, the standard enforced by the TCK guarantees only the lowest common denominator, and it's possible for a MIDP implementation to pass the TCK tests and not perform well at all: in particular, the TCK has nothing to say about how UI components appear on screen, or how user actions are transmitted to applications.

The TCK also allows a great deal of latitude in how applications are deployed and managed, and an additional problem is that despite the presence of device emulators, these PC-based emulators often perform differently from their real-world counterparts – especially if the actual phone is still in development.

Example Variations

Table 1 shows the kind of differences that can be expected between various MIDP platforms.

Let's examine some of the more serious differences. First, two issues about the environments into which MIDlets are deployed.

Releasing Resources

The J2ME specification states that a MIDlet may pass through three stages during its life, from creation in the

Severity	General	Management	Appearance	UI–Displayables	UI–Graphics
Major	A MIDlet may be required to release all its resources on a task-swap or it may not	Certain platforms can reject valid application installation files	A gauge showing value 0 may be visible or not	Gauges may display absolute values or percentages Application specified commands may appear on drop-down menus, as buttons, on full-screen menus	Colour depth can range from 1 bit to 16 bit Screen aspect ratios can vary from 3:2 to 0.5:1
Minor	-	The installed "application" might be one MIDlet, a suite of MIDlets or all the MIDlets on the device	Labels are not visually distinguished from content	In a Form, it may not be clear which editable item has focus	-
Trivial	-	-	Ticker can be at the top or bottom of the screen	-	-

TABLE 1 Differnces between MIDP platforms

Don't Miss Next month's *Java & Linux Focas Issue*

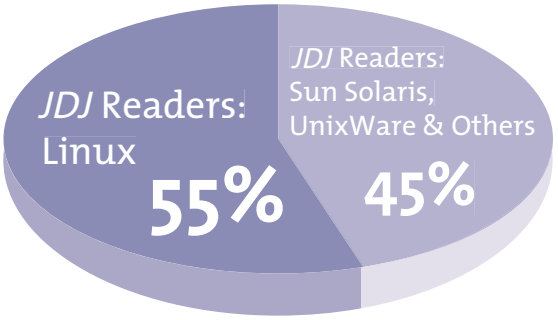
Over 70,000

Untapped Linux Professionals you're not reaching today!

The best-kept secret in the Linux community!

Java Developer's Journal reaches over 70,000 Linux developers, hardware and software buyers, and decision makers you're not reaching via any other publication in the market today!

JAVA DEVELOPER'S JOURNAL



Number of *JDJ* readers who are professional Linux developers

Bonus Distribution at LinuxWorld

LINUXWORLD CONFERENCE & EXPO

Call Today!

Carmen Gonzalez
(201) 802-3021
carmen@sys-con.com

Please call for deadlines.

Don't Miss* JDJ's *Linux Focus Issue!

With a total reach that's more than *Linux Journal* & *Linux Magazine* combined!

SYS-CON MEDIA

PAUSED state, through the ACTIVE state, to the DESTROYED state. The MIDlet class constructor can allocate any resources used only by the MIDlet, and startApp() can allocate "shared resources," which should be released when the MIDlet is paused. Nonshared resources must be released by destroyApp(). The intention here is that MIDlets PAUSED by the device's operating system must release expensive or volatile resources. This facility is meant to be used when a MIDlet is moved to the background on a task-swap.

Unfortunately, platforms differ in their interpretation of this feature. Some platforms, in fact, call destroyApp() when the MIDlet goes to the background, and create a new instance when it returns to the foreground. Clearly, a MIDlet that expects to be PAUSED but not DESTROYED could malfunction severely when restarted.

Rejecting Installation Files

FIGURE 1
Form wih gauge

J2ME, like the Standard and Enterprise Editions, uses Unicode characters. Some platforms on which J2ME is installed do not. This results in some platforms rejecting application descriptors that, quite validly, contain characters outside of the 7-bit ASCII character set (for instance, ä).

FIGURE 2
Gauge: value 2

It can be worse than that. Some MIDP-enabled devices do not install from .jar and .jad files, but from proprietary binary formats created from the .jar and .jad files (one example is the RIM BlackBerry). Some MIDlets can't be directly translated into RIM .cod files because attributes in the .jad contain ASCII characters that are not valid in a project name in the RIM tool used to build .cod files.

Command Presentation

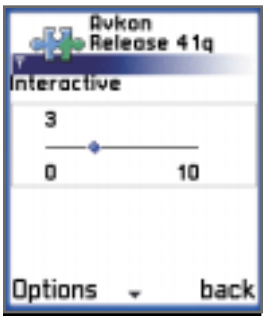


FIGURE 3
Platform displaying one command

Commands attached to a displayable will be presented to the user in the same way as commands that are part of the device's own UI are presented. This can cause confusion, since some platforms present commands in a menu that contains default items such as "select" and "exit". Some platforms present the commands in an immediate mode: simply selecting the command item sends an event to the application. On other platforms, a two-stage process takes place where the command is selected from a menu, then activated.

There are two problems here. Again, it's challenging to write user instructions that will make sense on every platform. Also, words belonging to a screen navigation metaphor, such as "select" and "back", should be avoided as names of commands. To a greater or lesser extent, the MIDlet's UI is presented within the context of the platform's own.

Trouble with Gauges

The gauge item illustrates some of the issues in UI appearance and interaction that occur with J2ME applications. On some platforms, the gauge item does not have a border, nor does it have any textual representation of the current value. So a gauge showing a value of 0 is invisible, merely a blank area on the screen.

On some platforms, a gauge item has accompanying text to indicate the current value, as well as the graphical display. On other platforms, this text displays the current value of the gauge and, still on others, it's the current value, as a percentage of the maximum value, that's displayed.

This uncertainty causes difficulties in writing portable applications using gauges, especially interactive gauges. Imagine also the difficulty in writing the user guide for an application using gauges, not knowing how the platform will display the gauge or its current value.

An Example

Figure 1 shows a form with a gauge, as implemented on Sun's reference platform. Figures 2, 3, and 4 show the same displayable from the MIDlet in the same state, but on three different platform-specific emulators. Notice that the reference implementation does not display any of the numerical parameters of the gauge, and neither does the platform in Figure 2, whereas the platform in Figure 3 shows all three parameters.

The platform in Figure 2 does display the title of the form being displayed ("Gauge: value 2") and displays both commands attached to the form ("back" and "continue"); the platform in Figure 3 does not display the form's title and displays only one command ("back"). The command "continue" is on the options menu, along with platform-supplied commands "copy text", "find in page", and "exit"". Both platforms indicate that the gauge has focus, by the ears shown on either side of it (see Figure 2), and the grey box surrounding it (see Figure 1).

On the reference implementation, it's the small arrows at the bottom of the screen that indicate that the gauge may be modified.

Contrast this with the platform shown in Figure 4. Here, the gauge shows the current and the maximum value (but not the minimum), and doesn't look much like a gauge at all. The gauge has focus, but in order to change the value, the "Edit" command must be chosen from the "Options" menu (see Figure 5) bringing the user to the gauge edit screen (see Figure 6). Clearly, any MIDlet will have very different usability on these three devices, but the one using gauges will provide spectacularly different user experiences.

FIGURE 4
Current and max value

FIGURE 5
Edit command

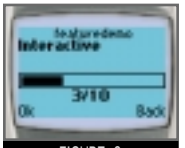


FIGURE 6
Gauge edit screen

Write Once, Run Everywhere

MIDP offers a highly uniform runtime environment across a wide range of mobile devices. However, differences between implementations mean that only the most trivial MIDlet is going to look the same on every platform.

As MIDP is deployed on more devices and a greater range of devices, from limited smartphones to powerful communicators, the true portability problem within J2ME – obtaining a consistent and consistently usable UI – will become ever greater. No two MIDP platforms, even from the same vendor, have exactly the same look and feel, so no MIDlet is ever going to have exactly the same user interface on every platform.

The most important MIDlet portability technique is to give up almost all the notions about user interface design that are so important in designing desktop or Web applications, and instead focus on giving your MIDlets as simple and small a set of user interactions as possible (with no assumptions made about appearance).

Emulators for all J2ME-enabled devices are available for (usually) little cost, so it's quite possible to exercise a MIDlet on many different platforms. This should be done by any developer who is truly serious about portability. Source code for this article can be downloaded from www.sys-con.com/java/sourcec.cfm.

• • •

White papers on the issues raised in this article are available from the Penrillian Web site, www.penrillian.com.

tonyr@penrillian.com

keith@penrillian.com

jezar@dreampoint.co.uk

Conference: January 21-24, 2003
Expo: January 22-24, 2003

The Javits Center
New York, NY

LinuxWorld Conference & Expo

Where OPEN Minds Meet

Linux is thriving! Financial services, telecommunications, government and retail are utilizing this amazing technology. Shouldn't you?

- Attend Keynotes and learn from inspiring visionaries who are devoted to Linux and Open Source.
  - Hector Ruiz, Ph. D., *President & CEO, AMD*
  - Jeffrey M. Birnbaum, *Managing Director & Global Head of Enterprise Computing in the Institutional Securities Division, Morgan Stanley*
  - Michael Tiemann, *Chief Technical Officer, Red Hat*
  - Steven A. Mills, *Senior Vice President & Group Executive, IBM Corporation*
  - Randy Mott, *Senior Vice President & Chief Information Officer, Dell Computer Corporation*

- LFS LINUX FINANCIAL SUMMIT

  All-new - Linux Financial Summit is an executive program created to address the growing demand for quality information and insight on Linux and Open Source solutions for the financial industry.

  Sponsored by: egenera, hp invent

- ENTERPRISE SOLUTIONS CENTER

  Check out the all-new Enterprise Solutions Center, where you'll experience multi-vendor Linux and Open Source enterprise solution demonstrations.

  Presented by: LINUX INTERNATIONAL, WILD OPEN SOURCE

- Participate in LinuxWorld's world-class education program and learn how Linux has been implemented in financial services, communications and government, gain technical and enterprise know-how in areas such as sys admin, business and emerging technologies.
- Explore the newest technologies from leading Linux and Open Source suppliers and a growing array of start-ups.
- Take your seat at The Golden Penguin Bowl, the lively quiz show.

www.linuxworldexpo.com

IDG WORLD EXPO

Cornerstone Sponsor

hp invent

Platinum Sponsors

AMD, IBM, intel, redhat, Sun microsystems — We make the net work.

Silver Sponsors

macrovision — Globetrotter Software Division

UNITEDLINUX

moving in new and exciting directions

Java and the Future of Ad Hoc Networking

written by Karl McCabe

In the dense forest of emerging computing trends, technologies, and hyped life-changing applications, there are two developments that stand taller than the rest. In isolation, these two trends are having a huge impact on users – both individuals and corporations. However, there is rich potential at the intersection of these two developments for a significant step forward in the way we use and implement computing devices and applications. These two trends are the ubiquity of mobile personal computing devices and the emergence of peer-to-peer computing architectures.

The proliferation of mobile computing devices hardly needs further documentation here – we're all aware of the sheer volume of mobile phones, PDAs, mobile pagers, and other assorted gadgetry that exists in the world. We watched as each "must-have" feature queued up to gain user acceptance. Some succeeded (SMS, e-mail pagers), some didn't (WAP, satellite phones), and for others the marketing guns are still firing in the battle for consumers (MMS, always-on networks, PDA phones). One thing is clear – devices with real computing power proliferate and are carried around by regular people. A top-of-the-range HP iPAQ (formerly Compaq iPAQ) PDA has 64MB of RAM and a 400MHz processor. This kind of hardware specification was typical for a PC purchased around the holiday season in 1999!

The emergence of peer-to-peer computing architectures has also been extensively documented. Systems such as Napster helped bring attention to the power of networks in which every node is a first-class citizen. This approach – building resilient, truly distributed systems that make effective use of processing power – is an excellent technical solution to difficult problems. More important, it can enable a genuinely new approach to application interaction. The mechanisms for discovering other nodes in a network, for interrogating those nodes to find out what services they provide, and for connecting to and using those services can all be revolutionized. The implications are far reaching, as evidenced by the visceral reaction of the music recording industry to Napster, et al.

The Java platform has an increasingly important role to play in many computing fields, not least of which is the field of distributed computing. In this article, I examine why ad hoc networks will be the key distributed systems paradigm in the near future – why they are in fact the first true distributed systems – and what role Java, particularly J2ME, will play in this new paradigm.

Why Is Ad Hoc Networking Useful?

Ad hoc networking can be defined as follows: *a network formed without any central administration, and whose nodes can dynamically, arbitrarily, and continually connect and disconnect.*

The single most important aspect of an ad hoc network is the absence of a central administration. This is more unusual than it sounds in a computing landscape dominated by servers – print, file, DNS, DHCP, application, Web, and so on. Taking the step to remove the central administration is both liberating and problematic. It gives us the freedom to create peer-to-peer networks on an ad hoc basis without the need for access points, base stations, or other infrastructure. On the other hand, we can no longer rely, for example, on central authorities to certify digital certificates – we must rely on some other mechanism to establish trust in an ad hoc network.

Our definition also points to other key aspects of ad hoc networks. The nodes must be able to connect dynamically – a network is not preformed and nodes can join and leave at any time. The nodes can connect arbitrarily to a network – that is, a node should not need (excessive) preconfiguration in order to take part in a network. Perhaps I'd like my PDA to form an ad hoc network with others that I meet at a conference.

Finally, ad hoc network nodes can continually connect and disconnect to and from a network – this is a natural thing for

International Call Sign for Projects in Distress:

.--- .- ...- .- / ... --- ...

(Morse code for "Java SOS")

Does your Java application crash and you just don't know why?
Does performance just not make the grade and support that demand?
Could your application just work a bit better?
Over budget already, and still have no clue how to complete all the requirements?
Been delivered a Java application you just aren't happy with?
Under pressure to deliver today but your Java application doesn't work?
Or just want some advice before you start developing?

if this sounds like you, then you need n-ary. We are recognized experts in the Java field, with in-depth knowledge in Java Architecture and Design. We're highly proficient in code and performance analysis, with a team committed to making your project a success. n-ary will put you back on track.

According to the Standish Group,

84%

of IT projects overrun their budget or fail. Don't let your project be one of them. Call us in before it's too late.

Headed up *by internationally renowned Java Expert, Alan Williamson, Editor-in-Chief of Java Developer's Journal and author of many Java books.*

n-ary

www.javaSOS.com

this network topology, and not an exceptional or error condition as is often the case in more sessile networks. Ad hoc network nodes are frequently mobile.

In addition to authentication, two other technical problems are accentuated in ad hoc networks – the "finding stuff" problem and routing. In any distributed system, a node needs to find another node that can provide some useful service to it. This is the "finding stuff" problem and it's usually solved with servers – DNS servers allow us to find Internet nodes using a friendly name; UDDI servers act as a Web-based repository of services. Without servers – i.e., in an ad hoc network – this problem is amplified. Likewise, routing information through an ad hoc network, particularly one consisting of mobile nodes, is a difficult problem. A node that's acting as a router might disconnect from the network at any time (it may go out of range). How does a network self-heal and allow seamless continuance of service in this scenario? Solving these challenges could yield a powerful and flexible network topology, a true distributed system.

There are many benefits that could accrue from such a distributed system. We could build networks more cheaply, because we don't need expensive server infrastructure. We could build networks that are more resilient. If each node acts as a peer, the network can heal as nodes drop out, thus providing increased quality of service. For users, simple ad hoc networks can form between their own personal devices – the so-called Personal Area Network (PAN) – or with nearby devices, and services may be accessed without a central infrastructure. I could share photos with a friend's device, pay for something in a vending machine using my phone, or buy a new stereo and have my universal remote connect automatically to it and configure it.

Ad Hoc Networking Technologies

Many physical networking technologies may be used to underpin an ad hoc network, though some are more suitable than others. Since nodes are frequently mobile, a wireless physical transport makes sense.

The wireless LAN technologies defined by the various IEEE 802.11 standards, also known as WiFi, are quickly becoming pervasive. The key protocols are 802.11b and 802.11a. The former operates in the Industrial, Scientific, and Medical (ISM) frequency band around 2.4GHz and delivers a transmission rate of 11Mbps. 802.11a gives a much higher data rate of 54Mbps and operates in the 5GHz band. In the second half of 2002, the vast majority of wireless LAN products on the market use the 802.11b standard. 802.11 slots neatly into the IEEE family of protocols as an alternative Media Access Control (MAC) layer; this means that things that work over Ethernet, such as TCP/IP, work with 802.11. This provides a reasonably seamless transition for existing applications and operating systems that need to "go wireless."

802.11 can operate either in Infrastructure Mode or Ad Hoc Mode. In the former, wireless nodes connect to a network through a WiFi access point, whereas in the latter, nodes may connect directly to each other without the need for an access point. Despite this, there are some aspects of 802.11 that detract from its suitability as an ad hoc networking technology.

As mentioned earlier, 802.11 is simply an alternative MAC layer for use over radio rather than wires. This benefit is in some ways a disadvantage though, since 802.11 networks tend to be just as ad hoc as fixed wired networks. Most of the protocols and applications that run over 802.11 networks are the same as those that run over wired Ethernet networks and are not particularly ad hoc (advances, such as Mobile IP, may improve this situation). Other downsides to 802.11 as an ad hoc networking technology include its relatively high-power consumption (a concern for small mobile devices, though less so for laptops) and the security problems that are beginning to be widely exploited. The 802.11 Wired Equivalent Privacy (WEP) security protocol has been largely discredited and does not provide adequate security. This may be a problem for ad hoc networks where "rogue" nodes may be within range. New developments, such as 802.1x and 802.11i, will address these issues.

The Bluetooth wireless standard, controlled by the Bluetooth SIG, also operates in the ISM frequency band. It's much less power hungry than WiFi, but it delivers a much lower transmission rate at 760Kbps. It was designed as a cable replacement technology and this is still its strongest role. However, it has certain characteristics that are ideal for ad hoc networking. Bluetooth supports device inquiry as a native feature, allowing a Bluetooth device to automatically discover other Bluetooth devices that are in range. This property makes Bluetooth well suited to self-healing networks or to systems containing nodes that want to detect and join networks in an ad hoc manner. Having discovered which devices are nearby, a Bluetooth application may then query the services that are supported by those devices. Using the Bluetooth Service Discovery Protocol, an application can search and browse the services provided by a remote device and query attributes about those services. These attributes can be used by the Bluetooth device to choose a suitable peer device and to configure itself to join the network.

Other physical transports worthy of mention here include Ultra Wideband (UWB) and proprietary protocols such as that used by Cybiko. UWB is inherently secure, requires very low power, and can achieve data rates up to 60Mbps. In February 2002, the Federal Communications Commission in the U.S. approved the commercial deployment of UWB (with limitations), and early development kits are now appearing. Cybiko is a wireless entertainment device aimed at young people and it supports multiplayer games, e-mail, messaging, etc. It's interesting from a technology perspective because it supports ad hoc routing. A device can communicate within another device that is out of range, as long as there's a path through one or more intermediate devices. This is a form of "mesh networking" for a specific problem domain – a topology that I'll return to later.

"The single most important aspect of an ad hoc network is the absence of a central administration"

Using Java to Build Ad Hoc Networks

There are a number of Java standards that are particularly relevant to the emerging ad hoc networking topology. The Java API for Bluetooth Wireless Technology (JABWT) is crucial since it maps Java software constructs onto physical ad hoc constructs. JXTA, Jini, and JavaSpaces are good examples of software frameworks that have a good architectural fit with ad hoc networks. Finally, I'll examine the new Real-Time Specification for Java – a key step in the proliferation of Java, and its success in new networking topologies.

JABWT

The Bluetooth specification is not an application programming interface (API)-based specification. Instead, it standardizes a set of protocol stack layers in such a way that the Protocol Data Units (PDUs) passed between stack layers and across the air are well defined and interoperable. This is perfectly appropriate for a wireless technology specification.

There are many implementations of the Bluetooth specification, or Bluetooth Stacks, available. Each of these stacks by necessity has its own proprietary API – usually a "C" API. An unfortunate side effect is the complexity involved in writing applications that use Bluetooth technology. A programmer's time will usually be taken up mastering the idiosyncrasies of the underlying Bluetooth stack API, figuring out how to initialize it, configure it, and coax it into providing the desired behavior. Some of the more mature commercial stacks provide some abstractions that aid usability, but these remain proprietary. The Java APIs for Bluetooth Wireless Technology (JABWT) for the first time provides a standard API for programming Bluetooth applications. Better still, this is a Java standard developed by the JSR-82 Expert Group. Motorola chaired this group, which also included Rococo Software, IBM, Nokia, and Ericsson. JABWT is an optional package layered on top of J2ME CLDC. Its current version is 1.0a, ratified in March 2002.

The JABWT APIs provide a number of key abstractions that simplify the tasks involved in building a Bluetooth application. In particular, they allow an application developer to initialize a stack, populate and read a service discovery database, perform device discovery, manage L2CAP and RFCOMM connections, manage Bluetooth security, and exchange OBEX data as efficiently and simply as possible. To achieve this, JABWT uses well-understood and widely used Java concepts, such as Listener classes for receiving asynchronous events and Universal Resource Locators (URLs) for identifying connection endpoints. The Input/Output (I/O) mechanism of the CLDC Generic Connection Framework (GCF) is extended by JABWT to include Bluetooth connection classes – RFCOMMConnection and L2CAPConnection. Since this is a standard networking framework used by all J2ME applications, programmers can quickly produce Java Bluetooth applications by applying existing techniques and design patterns.

This new standard is hugely important to the success of ad hoc networking. For the first time, it bridges the gap between a physical ad hoc technology, such as Bluetooth, and the richer, higher-level software frameworks provided by Java. It opens up the physical ad hoc capabilities of the new generation of mobile devices to the creative energies of the Java development community.

JXTA

JXTA (pronounced "juxta" from "juxtapose") was launched by Sun in April 2001. It's a programming and computing platform designed to solve a number of problems in the area of peer-to-peer networking. Peer-to-peer networking is essentially another name for ad hoc networking. It's a networking topology whereby end-user devices are connected directly to other end-user devices, without going through a third-party "server," for the purposes of sharing some information or services. When we talk about peer-to-peer, we usually don't expect the nodes to be mobile. Although many such networks have emerged, no interoperable, platform-independent platform has existed on which to build these collaborative applications.

JXTA is such a platform. It's an open-source technology that addresses such issues as peer discovery, peer information sharing, peer membership, and asynchronous pipes. It proposes an XML-based approach to advertising peer resources. JXTA is neutral to the underlying transport, though it seems clear that short-range wireless technologies such as Bluetooth could play a key role in its success. A device using JXTA over Bluetooth could discover other local JXTA devices and form ad hoc, peer-to-peer networks with them to achieve some common collaborative goal, or share information or services. The next year is likely to be critical in the evolution of the JXTA framework – it will be important that the technology starts to find its way into some real peer-to-peer solutions. Until then, the jury is out.

Jini and JavaSpaces

Jini, officially launched in mid-2000, is a technology that may be used to federate groups of users and resources required by those users. It aims to allow a network to be more flexible and more easily administered, with resources being easily located by humans and computational clients. Jini provides framework components for managing federated services, such as a lookup service for locating services, a leasing capability for controlling use of services and security, and transaction capabilities for coherent interservice interactions.

Jini operates at a level of abstraction that makes it attractive to application developers who need to solve an application problem, rather than worry about underlying networking issues. It provides a paradigm that may be useful to Bluetooth applications, since it allows services to be loosely coupled, that is, Jini nodes and services may appear and disappear from a network as a natural consequence of its operation. This loose coupling of entities is an important quality in a mobile ad hoc network.

JavaSpaces is a Jini service. It provides the abstraction of a JavaSpace into which components known as "entries" can be put. An application may "find" these entries, it can "read" (passive) or "take" (destructive) entries, and it may write entries to a space. The JavaSpaces architecture includes a notification mechanism that notifies an application when an entry is written that matches some criteria. This form of abstraction is particularly useful for passing information from stage to stage in a workflow model, or for applications where redemption of some token is important – for example, redeeming a voucher with a service provider. Interestingly, JavaSpaces also provides a useful abstraction for layering over actual physical spaces. In a Bluetooth access point scenario, this may be useful as an application model for mapping zones or spaces in which access points are installed, and for specifying which services are available in which zones.

Both Jini and JavaSpaces are good examples of creative and elegant designs for Java software frameworks that are intended to provide rich abstractions to application programmers. They have characteristics that make them well suited to ad hoc networking applications – particularly the inherently loose coupling of objects. They're not without their drawbacks, though, as I'll examine shortly.

"For Java to be a credible platform for building ad hoc networking applications, it needs to jump a number of technical hurdles"

Real-Time Java

The Real-Time Specification for Java was developed under the Java Community Process and was released in January 2002. It defines a set of extensions to the Java Language Specification, the Java Virtual Machine Specification, and an API that enables the "creation, verification, analysis, execution, and management of Java threads whose correctness conditions include timeliness constraints (also known as real-time threads)." The specification defines an environment that retains the key portability and memory management benefits of Java, while emphasizing predictable execution as the top priority. This specification is an important development in the ongoing effort to encourage the proliferation of Java technology.

A recent article by Jim Turley, "Embedded Processors by the Numbers" (*Embedded Systems Programming*, Vol. 12, No. 5), claimed that if you round off the fractions, embedded systems consume 100% of the worldwide production of microprocessors. Given the number of processors in the average home (household appliances, remote controls, stereo equipment, games consoles) and car, this claim is probably reasonable. If two such devices need to form a network to share a service, an ad hoc network may result – for example, a universal remote control might "discover" the new TV you just bought and allow you to control it remotely. If Java is to be a credible platform for ad hoc networking, it's imperative that it can migrate onto embedded microprocessors. The importance of the new Real-Time Specification for Java is clear in this context.

Java Lip Service

For Java to be a credible platform for building ad hoc networking applications, it needs to jump a number of technical hurdles. For starters, Java needs to work on small devices. This has been comprehensively addressed by now, of course, as evidenced by the widespread deployment of mobile Java by companies such as Siemens, Motorola, Aplix, RIM, and Sharp. As an aside, there's a significant danger of the mobile Java standards fragmenting due to proprietary additions being made by various vendors. This has been driven by market necessity – the Java Community Process has struggled to keep pace with the shortened design cycle of modern gadgets. New standards, like MIDP 2.0, and "oversight" groups, such as the JSR-185 Expert Group, should help to address this problem (JSR-185 EG is tasked with providing an architectural description, reference implementation, and compatibility framework for "Java Technology for the Wireless Industry").

For ad hoc networking, though, it's not enough to be small. There are certain things that it's hard to live without when building full-featured distributed systems – Remote Method Invocation (RMI) and the Java Native Interface (JNI), both missing from J2ME's Connected Limited Device Configuration (CLDC), are chief among these. The absence of RMI is explained by the memory constraints of CLDC, which is reasonable. The downside is that "traditional" (i.e., non–Web service) applications involving rich object interaction are difficult to build; HTTP is not always a suitable transport for this type of application.

The absence of RMI has another impact that is perhaps less forgivable – Jini depends on RMI. Jini doesn't work with CLDC. This is a pity, since CLDC devices are exactly the kind of devices that could benefit from Jini's lookup architecture and code mobility. I'd like my mobile phone to be able to discover the nearest printer, query it to get a print driver, and then print my phone list or WAP page. If my phone runs CLDC, this isn't possible (though companies like Psinaptic are working to change that). Sun Microsystems are now positioning Jini as an architecture for building network-centric services, moving away from the promise of Jini in gadgets.

The omission of JNI from J2ME CLDC does not have the same direct impact on J2ME application programmers. The reason for its absence is, again, sensible. The CLDC security sandbox model requires that "the set of native functions accessible to the virtual machine is closed, meaning that the application programmer cannot download any new libraries containing native functionality or access any native functions that are not part of the Java libraries provided by CLDC, profiles, or manufacturer-specific classes" (from the CLDC specification). The implication of this for developers, though, is that they cannot access underlying native protocol stacks, such as Bluetooth, from Java. A developer may want to do this in order to access features that are integral to ad hoc networks – for example, to discover nearby Bluetooth devices and what service they provide.

Crystal Ball

The future of ad hoc networking is likely to be determined by the emergence of compelling use-cases. If ad hoc networking leads to faster, cheaper, or easier ways of carrying out tasks, or if it leads to new revenue opportunities for companies, then it will succeed as a topology.

Mesh networking seems well placed to satisfy some of these criteria. A mesh network is one in which every node may act as a router and repeater for every other node in the network. Taking this seemingly simple step allows true ad hoc networks to form wherever there are sufficient nodes. No infrastructure is required – the nodes are the infrastructure. In a busy urban environment, voice and data traffic could be routed from one device to another, even if those devices were not in range of each other, by hopping from device to device. This approach will probably either revolutionize the way networks work, or it will be buried by the infrastructural vested interests, such as wireless voice and data carrier companies.

Another development to watch is the likely spread of "residential gateways." Companies that already have equipment in consumers' homes – such as set-top-box manufacturers – are planning ways to extend the reach of functionality of such equipment. One way to do this is to allow set-top boxes to establish ad hoc networks with other equipment in the house (entertainment equipment, phones, children's toys, universal remotes, PCs, etc.). An ad hoc network that includes a set-top box connected to a high-bandwidth connection provides rich potential for new application types. Java will play an important role in this development, as it is central to emerging standards in the Digital TV area such as the Multimedia Home Platform (MHP) and the Open Cable Applications Platform (OCAP).

These developments and others that are beyond the scope of this article, such as JavaCard and OSGi , are pushing computer networking in new and exciting directions. Decentralized topologies are sure to be at the forefront and ad hoc networks built on Java infrastructure and running Java applications can play a crucial role.

Author Bio

Karl McCabe is chief technology officer at Rococo Software, a Dublin-based wireless Java company. Karl was formerly responsible for the development of the Java product line at IONA Technologies and he has worked in various technical roles at IONA and ICL. He has helped to build and deploy distributed systems for many Fortune 500 companies, and his current interest is in extending the reach of these systems out to mobile devices.

karl.mccabe@rococosoft.com

Adalon 2.2

by Synthis Corporation

REVIEWED BY CARL BURN cburn123@yahoo.co.uk

Synthis Corporation
430 Tenth Street NW
Suite S-108
Atlanta, GA 30318
Web: www.synthis.com
E-mail: info@synthis.com
Phone: 1 866-SYNTHIS

Minimum Requirements
OS: Windows
Processor: Intel PIII/AMD K6 550MHz
Memory: 128MB

Test Environment
OS: Windows 2000 Pro
Processor: 800MHz Intel PIII
Memory: 256MB

Info:

Is there a development team out there who has never heard the following phrases when delivering a project? "That's not what we wanted…" or "We thought it would also do this…" Then even when we are on target, we're all too often hit with requirements that were simply not communicated and end up failing to meet our client's expectations.

Adalon 2.2

This is a new type of tool specifically created to enable developers to accurately define and design complex Web-based software applications from a business/functional perspective.

The idea here is that we build a model of the application before we go to code and keep all the requirements and modeling within the bounds of one integrated design tool.

Installing and Using Adalon 2.2

Adalon is available for purchase from the Synthis Web site or from a local partner. A demo copy is also available from the site following a simple registration form; activation keys are simply e-mailed to you.

Installation was painless and the application ran easily on an old laptop with limited resources (see Figure 1).

Using Adalon is also relatively simple. Using the drag-and-drop interface you can put together a diagram of the application, and you can use several diagrams for complex applications.

The diagrams are backed up with forms that gather further information and flesh out the details of the application (see Figure 2).

Adalon generates a board range of design deliverables, including standard functional and technical design documentation, an HTML-based click-through model of the application, and the framework components for the application.

The output components are determined by the "skin" you choose. Using the Struts skin, the major elements of the application structure were autogenerated at three levels:

1. *At the framework level*, where all Adalon design elements and generated files map closely to standard Struts framework elements
2. *At a slightly higher level of abstraction*, by using some common, open-source Struts extension packages to generate everything needed to build, deploy, and test the Struts application
3. *At the enterprise level*, by using a light processing model and the XDoclet Javadoc doclet engine to create application server–specific EJBs

Adalon doesn't try to generate your complete application. Instead, it generates an accurate translation of the business functional designs into a high-level code framework that supports a team's particular architecture. Generated code artifacts include Struts configuration files, base Struts classes, and JSP templates.

Adalon is available for developers who are

ATTN:
Wireless Developers...

WDJ
JAN'03

Coming in January:

SYS-CON Media's NEW, EXPANDED COVERAGE of wireless development...and developments in wireless

▶ ***WDJ*** focuses on wireless solutions from the perspective of everyone from senior IT management to the individual wireless developer, both corporate and independent.

▶ ***WDJ*** features regular monthly columns by the world's top developer-writers on the **major** wireless platforms, the **major** wireless hardware, and the **major** wireless applications... with the accent firmly on drilling down into every aspect of Wireless That Works.

▶ Bill Ray, ***WDJ***'s Editor-in-Chief, has been developing handheld applications for 16 years – from a mobile data gathering system deployed in 1986 to Bluetooth games still in development. He has worked with every category of application, including the world's first Java-capable DTV set-top box, which he developed for Swiss Telecom.

Bill Ray
Editor-in-Chief

Wireless Solutions That Work!

Exclusive Developer Coverage

Look for specific "developer" columns covering:

DEVICES:	***PLATFORMS:***	***TECHNOLOGIES:***
Palm / Handhelds	*Palm OS*	*Bluetooth*
HP / iPAQ	*Symbian OS*	*802.11*
Sharp / Zaurus	*WinCE*	*SMS / EMS / MMS*
RIM / BlackBerry	*J2ME*	*SIM Phase 2+*
Dell / Axim X5	*BREW*	*and more...*
Nokia / Communicator	*.NET Compact Framework*	
and more...	*Linux*	
	Nokia Series 60	
	and more...	

SUBSCRIBE TODAY!

1-888-303-5282

WWW.SYS-CON.COM

SPECIAL LIMITED TIME OFFER

ONLY $49.99

FOR 12 ISSUES

COMING
JANUARY '03 ▶ wireless DEVELOPER'S JOURNAL

SYS-CON MEDIA

using a number of platforms and methodologies. In the Java arena, the Apache/Jakarta Struts framework is fully supported and the terminology of Struts development is built into the application in the form of an optional GUI skin. The Fusebox framework, used by Web developers working in JSP, CFML, and PHP, is also supported as standard within the product.

If you use a customized version of Struts or Fusebox, another third-party Web architecture, or an internal proprietary standard, Adalon's code and documentation generation capabilities can be easily customized to transform Adalon's XML-based output into whatever format you need.

Summary

Synthis has come up with a nice tool. It's easy to come to grips with it, which is more than can be said of some other application modeling tools.

Its not all plain sailing. Adalon has a learning curve and like all new things, it requires effort to get into the habit of using it and encouraging others to follow suit. The only gripe we could come up with is that the tool is not structured for multiusers; a team version with a source control of models would be nice.

• • •

Note: Since this product review was written, a new version, Adalon 2.5, has been released.

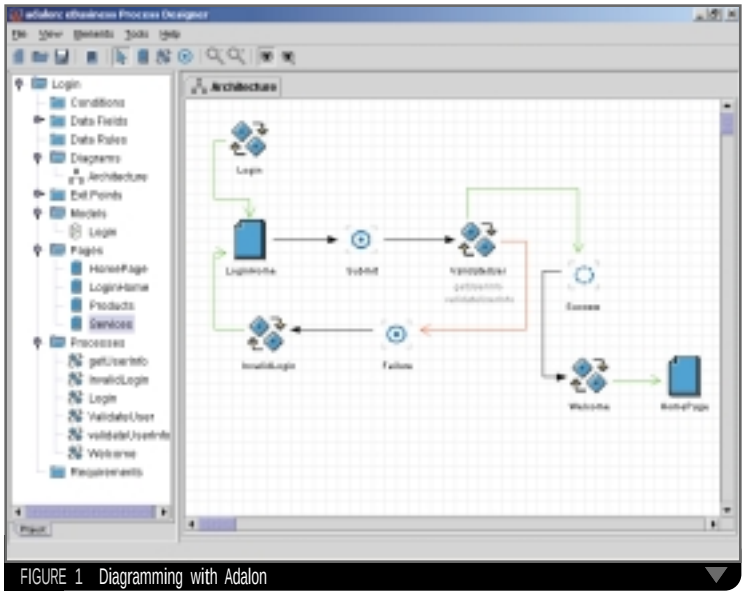


FIGURE 1 Diagramming with Adalon

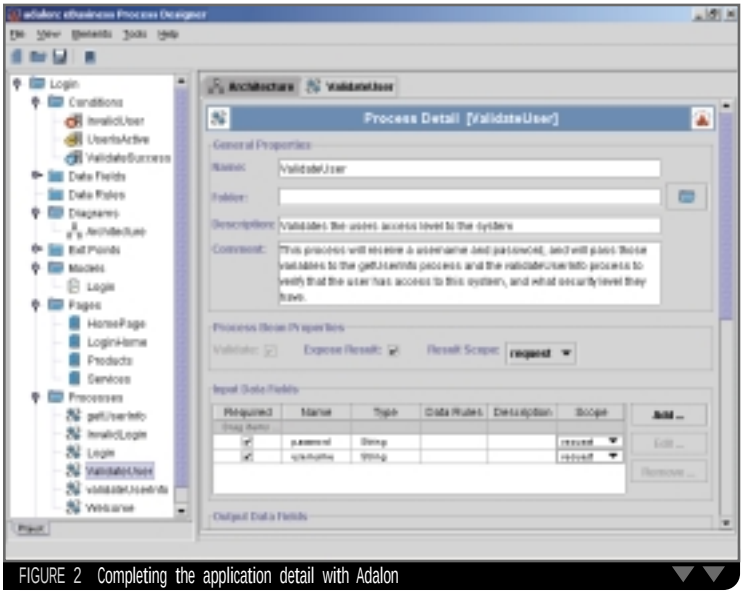


FIGURE 2 Completing the application detail with Adalon

Adalon Snapshot

Target Audience: Business analysts, Web developers, and Java developers

Level: Beginner to advanced

Pros:

- Easy to use
- Creates documentation as part of the modeling process
- Multiple output formats
- Can be customized using XSL

Cons:

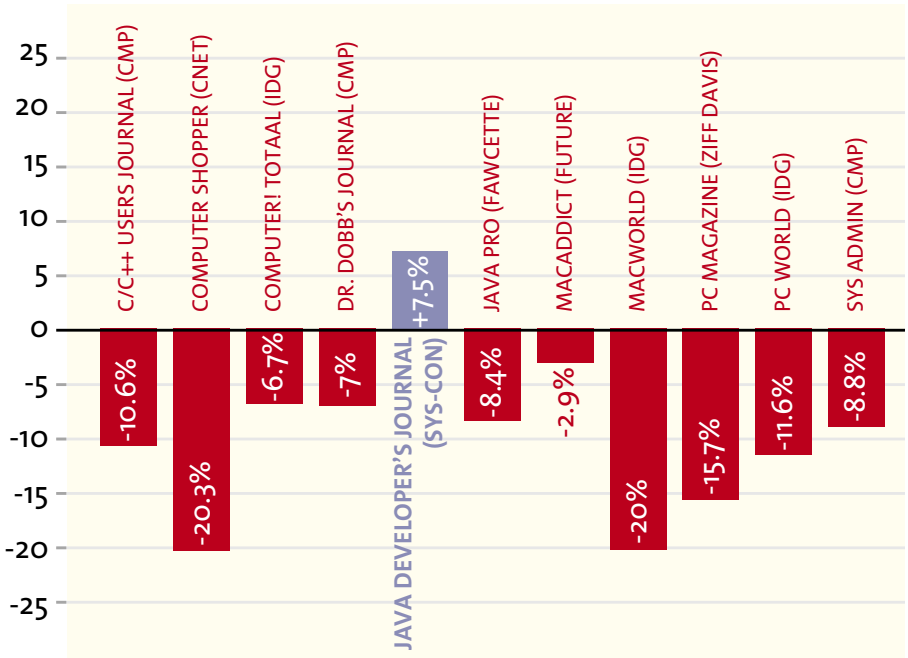
- Nothing significant

#1

i-Technology Magazine

in the World!

MAGAZINE SALES TREND 2002 VS 2001



C/C++ USERS JOURNAL (CMP)	-10.6%
COMPUTER SHOPPER (CNET)	-20.3%
COMPUTER! TOTAAL (IDG)	-6.7%
DR. DOBB'S JOURNAL (CMP)	-7%
JAVA DEVELOPER'S JOURNAL (SYS-CON)	**+7.5%**
JAVA PRO (FAWCETTE)	-8.4%
MACADDICT (FUTURE)	-2.9%
MACWORLD (IDG)	-20%
PC MAGAZINE (ZIFF DAVIS)	-15.7%
PC WORLD (IDG)	-11.6%
SYS ADMIN (CMP)	-8.8%

DATA: CURTIS CIRCULATION 2002 TREND ANALYSIS REPORT. SINGLE COPY SALES TREND, PERCENTAGE CHANGE FIRST HALF 2002 VS 2001. CURTIS CIRCULATION IS THE WORLD'S LARGEST MAGAZINE DISTRIBUTOR.

Miles Silverman
VP Marketing

Carmen Gonzalez
Senior VP Marketing

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

JAVA DEVELOPER'S JOURNAL

Rational XDE

by Rational Software Corporation

REVIEWED BY JASON BELL *jasonbell@sys-con.com*

Rational Software Corporation
18880 Homestead Rd.
Cupertino, CA 95014
Phone: 800 728-1212
E-mail: info@rational.com
Web: www.rational.com\tryit\xde\javaed.jsp#dl#dlwsw

Specs
Price: $3,595
Platforms:
Windows NT (Service Pack 6a)
Windows 2000 (Service Pack 2)
Windows XP Professional

Test Platform
Computer: Dell Optiplex GX240,
Processor: P4 1.8GHz
Hard Drive: 40GH
Memory: 256MB RAM
OS: Windows 2000 (Service Pack 3)

Over the past few years the integrated development environment has raged on. For years I've used a text editor and a build tool to create my Java software; I used no set processes or design methods. The integration of UML modeling and Java coding makes obvious sense to a software designer, but a programmer like me has to ask, can XDE mentor me and make me think differently about the way I design software?

XDE stands for e**x**tended **d**evelopment **e**nvironment; installation is a straightforward matter, as is installation of software on most Windows systems, but it only runs on Windows 2000/NT/XP and you'll also need Service Pack 2 installed. The entire suite is based on IBM's Eclipse Platform but with added functionality in the UML modeling department. The distribution comes with two CDs – one that installs XDE in IBM's WebSphere Application Developer (WSAD) and one that installs XDE "standalone" with its own copy of WebSphere Workbench (WSW), which is based on Eclipse. For this review I just installed the stand-alone workbench.

Getting Started

Before I could get any coding done I had to create a project; the Java Modeling Project has the full features of UML modeling and round-trip engineering support. There are other project types you can use, including a Java Project (no modeling support) and a Web Modeling Project (for JSP or HTML Web applications).

Using XDE

The bulk of the modeling and coding goes on within the modeling perspectives. Rational has provided a lot of information for the help perspective covering Java and UML basics, deploying applications, and using patterns. One of the joys of XDE is that you can let it do as little or as much as you like. The level of code assistance is controlled from the preferences, and there are various options that can be generated such as constructors, finalizers, and bean properties. With synchronization enabled you can alter the source code and the UML diagram will update; if you make changes to the diagram the code will update accordingly (see Figure 1).

A very handy feature is the task list where you can add items for anything that you need to complete. This is also the place where any compilation errors are listed; clicking on an error will take you to the corresponding line of code where the error is. Once corrected, XDE will automatically update the task list and delete the corrected errors from it.

You can create and place new classes on the main model diagram either by using the right-click menus or by dragging and dropping from the Toolbox. The Toolbox is home to the different UML types (class, component, deployment, use case, collaboration, and sequence diagrams) plus Java elements (class, methods, interfaces, etc.), EJB elements (session, entity, and message-driven beans), and JSP elements (servlet, JSP server page, HTML forms, etc.). Features on the modeling side include Gang-of-Four patterns, J2EE patterns, and the ability to create your own patterns and code templates so you can reuse your core code over a number of projects.

Developing for the Server Side

As I mentioned previously, you can develop JSP/servlet applications with XDE. With a mixture of drag-and-drop elements you can quickly create a basic Web application and deploy it to your application server. Deployment is achieved with the help of a wizard that guides you through the process (see Figure 2). XDE has a number of server brands that you can deploy to, including WebSphere and WebLogic. If your server is not

listed you can deploy to a .war file and change the descriptors within XDE. This is a handy route if you want to deploy to servers such as Tomcat or Orion.

Handy Features

Where do I start? I'm a big fan of code formatting, and XDE has the ability to do this with its menu within the edit window. All you have to do is right click, click on "Format", and it's done. There are various styles of code layout that you can configure within the preferences. Another handy feature is assisted coding, in which XDE inspects the signature of a class that you're working on and lists its different types. I found this handy, mainly because while I was working I didn't have to stop, open up *Java in a Nutshell*, and then continue where I left off (see Figure 3).

In the main preferences you can specify which JDK version you want to work with. I found this very helpful, as I was writing and testing between JDK 1.3 and 1.4. I could switch mid coding and see if there were going to be any issues or conflicts between different Java versions.

Code Refactoring

Have you ever developed code and then changed the name of something and then systematically had to revisit every file to make the changes? There are numerous ways of making the change (mine was always using Perl on the command line). With XDE you can refactor code and it will make the changes across the project. XDE will let you preview the code results it is going to refactor, which gives you a fighting chance if you disagree.

One thing about Rational tools is that they are designed to work with other Rational tools. XDE links into ClearCase so you can have proper version control, but it gives no ability to use any other version-control software, such as SourceSafe or CVS.

Debugging

There are extensive debugging options, with the usual breakpoints, that keep an eye on variables. XDE will scan the packages open and give you the option of executing any class that it can. In addition to debugging on your local machine you can debug a class remotely on another machine or server – as long as it has the relevant profile set up so it can be debugged by XDE.

Summary

I've only scratched the surface of what XDE is capable of, but I have definitely increased my knowledge of software development through using this product. It's now part of my main development arsenal, and I'm finding myself actually thinking differently about software design and logically modeling objects before I do any coding. I know it sounds like a cliché, but I really don't know how I got on without it.

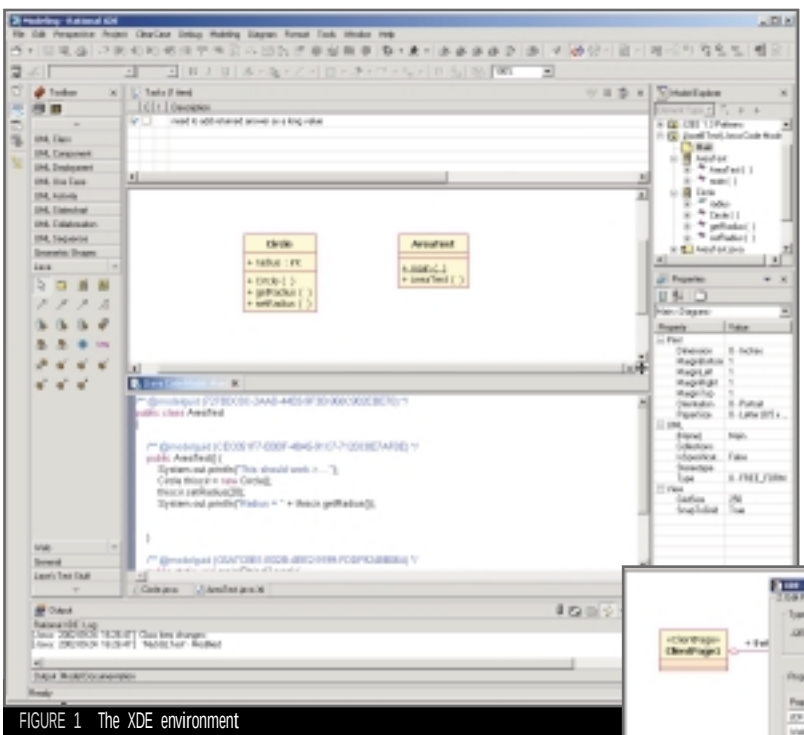


FIGURE 1 The XDE environment

FIGURE 3 Code assistance

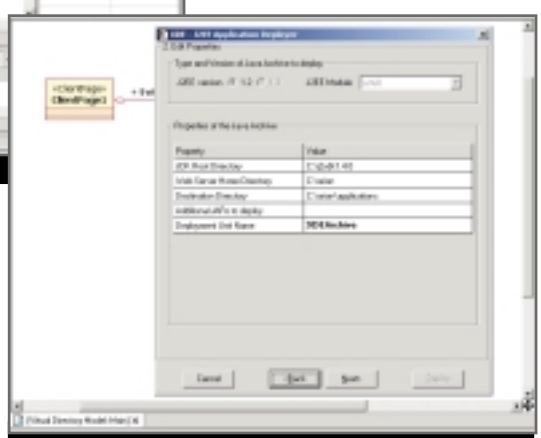


FIGURE 2 Deploying Web applications wizard

JDJ Product Snapshot

Target Audience: Java programmers, software designers, and analysts

Level: Intermediate to professional

Pros: Code/model synchronization

Ability to select different JDK during development for testing. Extensive coverage of patterns and software design practices.

Cons: Help system can be confusing at times; price may put some users off

COMDEX Opens with Largest Drop in Attendance Ever

In the world of computing, it's well known that everything is supposed to get smaller…it's just that no one ever thought it would one day happen so dramatically to COMDEX, once one of the world's largest technology shows.

According to an informal poll conducted among the very informed Las Vegas cab drivers, Sheldon (Shelly) Adelson, the owner of the Venetian Hotel (www.venetian.com) in Las Vegas, and who originally sold The Interface Group and The COMDEX Trade Shows to SoftBank, may be in line to buy the show back and keep it in Las Vegas. Sources close to Mr. Adelson denied the rumor. So far, no official news on the anticipated bankruptcy filing for the show organizer Key3Media, and the outcome this step will bring to the table.

Grisha Davida, vice president of SYS-CON Events, said he was exploring the possibility of moving SYS-CON's West Coast event to Las Vegas during COMDEX week in 2003.

Las Vegas Cabbies and COMDEX

"As usual, the best sources of information about Vegas are the cabbies," says Jim Louderback of TechTV. "When I arrived, my cab driver told me that his buddy was carting around a Key3 exec who was complaining about the scanty turnout – only about 75,000 had preregistered, less than the 125K that COMDEX claims will be here, and well below the 225,000 or so of COMDEX 2000. My driver today passed me a copy of the Las Vegas Convention Calendar 2002–2006. COMDEX, alas, was missing from 2003, 2004, 2005, and 2006. His inside dirt: the show moves to Houston next year."

Our experience with our cabby was a little more involved. "COMDEX is moving to Florida next year." He continued, "Are you a big shot, sir? If you are, please tell Scotty that I need to speak to him just one minute on the phone. I have the patent for the Java Cabby, which will make him billions of dollars. Right now the car is working on Windows; he has to talk to me and we will make this a Java car."

Our cabby told us that he was at home working on the car during his lunch break and swore that his Java Cabby would be a billion-dollar business (www.emcz.com).

"Everything Is Possible"

Back at the show, Hewlett-Packard CEO Carly Fiorina in her keynote unveiled HP's newest advertising slogan – "(Customer) + HP = Everything is Possible". It will be used by HP to help underline Fiorina's newest message, which is – basically – that HP is a "full-service" computer company offering end-to-end solutions.

"While the demand for technology for its own sake may be down," Fiorina stated, "the need for technology that solves real human and business problems is stronger than it has ever been."

Carly Fiorina

McNealy's "Third Wave" Washes Over COMDEX

Hall C5 of the Las Vegas Convention Center was definitely the place to be at noon on Monday...because Sun's chairman, president, and CEO Scott McNealy was in Las Vegas to give his keynote address.

McNealy discussed the state of network computing and the IT landscape. Trends in network computing, he asserted, dictate that we all start paying particular attention to developments on the client side and the ongoing importance of open interfaces and industry standard protocols as the world moves further into what he called "the third wave of network computing."

Scott McNealy

SYS-CON Publisher Denied Press Badge

The COMDEX Media Registration manager on site refused to issue a press badge for SYS-CON Media group publisher Fuat Kircaali, even after Kircaali presented his press credentials, photo ID, and business card. Kircaali was covering COMDEX for SYS-CON's 10 leading *i*-technology magazines. The Key3Media employee who introduced herself as the manager in charge of press/media registrations said, "We know SYS-CON Media and we know you, however, you need to show us your press credentials and bring us copies of your work. We can not issue a press/media badge for you at this time."

Reader Feedback to COMDEX Report

There are two main reasons why COMDEX is failing:

1. *The Web has largely made technology shows obsolete.*
2. *There's nothing new out there. The latter can be squarely blamed on the fact that the computer industry has become hostile to new ideas. Everything has to be the same old humdrum PC stuff. Ten years ago, everyone was trying out new ideas. Nowadays, nobody wants to try anything new because there's no hope of making any money.*

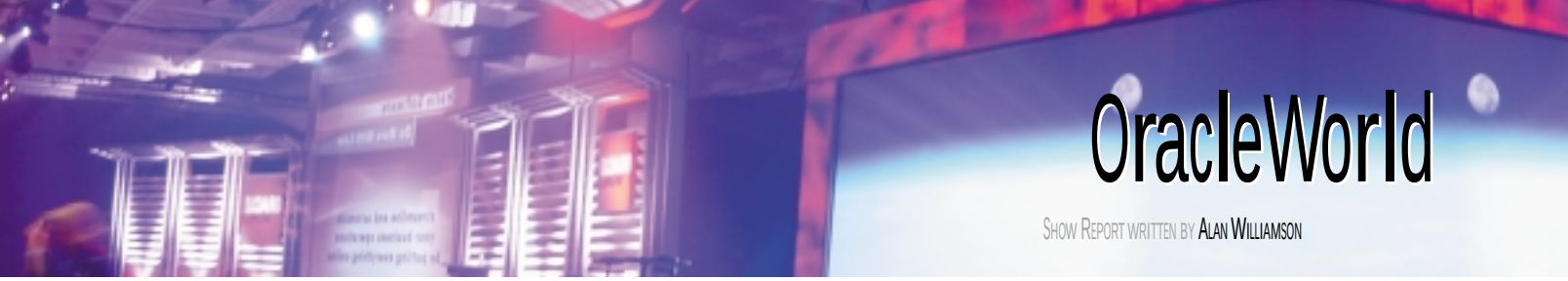
Is it any wonder that the only part of the tech world where change is happening is in the open source universe? Where the business rules don't apply?

via e-mail

It's not just one thing that is causing COMDEX to decline; it's a combination of things. The computing world is so diverse now that no single event could even begin to cover it all. That, combined with the simple fact that innovation has moved away from the MS and PC platforms means that we now have to choose where we go based on our special needs and interests.

In the '90s the PC world was pretty simple but in 2002 just the PDAs alone could form a convention! Add to that the special needs of data center people (another convention there, for sure), various server-based architectures, and engineering/scientific types and you have three "shows" where before COMDEX tried to fill all these niches (or the niches didn't even exist).

craig@networkessentials.net



OracleWorld

Show Report written by Alan Williamson

On November 10–14, San Francisco played host to Oracle's flagship show OracleWorld. Oracle, a company renowned for not doing things by half, put on the sort of show that hasn't been seen since the heyday of the dot-com era. It had all the ingredients that a tech show of this nature should have: great keynotes, great sessions, and a flamboyant show floor.

Michael Dell

The main theme of the show wasn't, ironically, spend-spend-spend but save-save-save. The focus was on how Oracle, if deployed, could deliver not only a cost-effective solution by shaving off large percentages of your current IT expenditures, but also deliver this infrastructure with the now infamous "unbreakable" mantle.

Unbreakable, eh? Marketing hype or fact? At **SYS-CON Radio** we interviewed many of the top people at Oracle and put this very question to them (www.sys-con/java/oracleradio/radio.cfm). The resounding answer was that it was fact and being practiced in every step of the Oracle process to ensure that no department let the overall team down – an admirable philosophy. Oracle continued to boast how after over a year of preaching this, not one person had managed to successfully claim the bounty that was offered if they could prove it "breakable."

As you may or may not be aware, Oracle is more than just a database company. They have been working hard over the past few years to reposition themselves as the complete solutions company, and for the outside world to see them as the one-stop shop for all their IT requirements. An example of this can be seen in how they promote their Collaboration Suite, in particular their e-mail servers. In his keynote, Larry Ellison commented on how Oracle has reduced its 90+ e-mail servers down to one using the Oracle suite; the quintessential example of a company eating in its own kitchen.

OracleWorld lined up top executives for a very impressive keynote roster: Michael Dell (Dell), Paul Otellini (Intel), Carly Fiorina (HP), John Gage (Sun) and, of course, beamed in live from New Zealand, Larry Ellison. The Dell keynote was of particular interest, as it was a pure product pitch on how Dell servers with RedHat were slowly dominating the server market, complete with sound bites from Ellison proclaiming that Oracle was now running all their servers on Dell. Again the resonating theme was overall cost reduction, which could be accomplished by employing the killer combination: Dell + RedHat + Oracle.

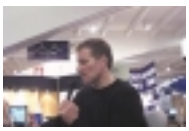
The overall theme of the exhibit floor was on the exact same topic: we can save you money, we can give you that all-important return on investment.

The tradeshow floor was large, and it took a long time to cover it all. Unlike many conferences, there was a lot to do and hear at various exhibitor stands, including sitting down and listening to some rather large in-stand presentations. Vendors didn't stand up and talk; they put on a show, cheering and whipping up their audience in a shout-aloud frenzy. There was energy to be had from the show floor, an energy that hasn't been experienced for many a JavaOne now.

OracleWorld, a show that isn't normally on my conference schedule, impressed me and I shall be making a point to catch it next year.

SYS-CON Radio Host Alan Williamson Interviews...

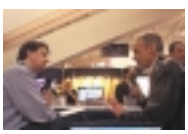
Ted Farrell, Architect and Director of Strategy of App Development Tools for Oracle



John Entenmann, Vice President of Business Intelligence Tools for Oracle

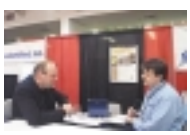
John Magee, Vice President of Oracle9*i* Product Marketing

Steven D. Levine, Vice President of Marketing Collaboration Suite for Oracle



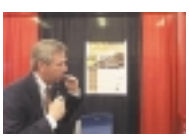
Benny Souder, Oracle Grid Computing Guru

Jacob Christfort, Vice President of Mobile Services for Oracle



Rene Bonvanie, Vice President of Oracle9*i* Product Marketing

Christopher Job, Vice President of Application Development Tools for Oracle



Robert G. Shrimp, Vice President of Oracle Database Product Marketing

HiT Software DB2Motion Adds Oracle9i and MySQL Support

(*San Jose, CA*) – HiT Software has announced the availability of DB2Motion v3.0, which adds Oracle9*i* and MySQL databases to its set of supported platforms. These newest database platforms add to existing SQL Server, Oracle, and iSeries support. Using DB2Motion, iSeries and AS/400 administrators can define bidirectional data replication while optionally extending detailed control via standard VBScript. By adding support for the latest Oracle release and popular open source MySQL, iSeries users can confidently channel data to the most popular database platforms for optimal application access.
www.hitsw.com

Rococo/AI Corp Distribute Embedded Java Solution in Japan

(*Dublin, Ireland*) – Rococo Software and Japan-based software distributor AI Corporation have announced their partnership to promote and distribute Rococo's embedded Java solution in Japan. AI Corporation will distribute Rococo's products and services aimed at making it easier to build Bluetooth applications in Java.
www.rococosoft.com

QUEST SOFTWARE TO ACQUIRE SITRAKA

(*Irvine, CA / Toronto, Canada*) – Quest Software, Inc., a provider of application management solutions and Sitraka Inc., a leader in application-server performance management, have signed a definitive agreement for Quest to acquire Sitraka. The acquisition broadens Quest's application management portfolio of products to enable customers to detect, diagnose, and resolve application bottlenecks within multitiered Java application environments.

Under the terms of the purchase agreement, Quest will pay $51.7 million in cash and up to $3.0 million in additional contingent consideration.
www.quest.com
www.sitraka.com

WebAppCabaret Offers Complete J2EE and EJB Hosting

(*Santa Cruz, CA*) – WebAppCabaret, the number one J2EE hosting and outsourcing company now offers a variety of hosting plans based on the Java J2EE framework. Hosting plans include support for EJBs, servlets, and JSPs.
www.webappcabaret.com

Compuware Accelerates Development of Java Applications

(*Farmington Hills, MI*) – Compuware Corporation has introduced DevPartner Java Edition 3.0, a suite of software development productivity tools that speeds the delivery of Java applications and components. Accessed through a Web browser interface, DevPartner Java Edition uses analysis and profiling techniques to help Java developers understand how their code performs before an application is deployed. This helps companies decrease time-to-market, reduce development costs, and increase developer productivity, ensuring the development of high-quality Java applications.
www.compuware.com

Motorola Implements to Open Standards

(*Libertyville, IL*) – Motorola, one of the world's largest providers of Java-enabled handsets, has announced that it is enhancing its open standards–based portfolio to operators around the globe through a series of agreements and offerings that will pave the way for more personalized mobile communications experiences based upon J2ME technology. These agreements will allow Motorola to enhance its end-to-end J2ME leadership by expanding mobile entertainment and productivity services for mobile phone users and providing Web-based training for software developers interested in writing code for Motorola's Java technology-enabled phones.
www.motorola.com

webMethods to Integrate JBoss J2EE-Based Application Server

(*San Francisco*) – webMethods, Inc., a provider of integration software, has announced that they are incorporating the award-winning JBoss application server directly within the webMethods integration platform, marking a revolutionary shift in the enterprise software market. This combination of technologies gives customers the freedom to combine business and integration logic directly within the webMethods integration platform. From a single integration platform optimized for Web services and extended enterprise integration, users can exploit the strength of the J2EE-based development paradigm to complement their integration logic.
www.webmethods.com
www.jboss.org

APPLE RELEASES WEBOBJECTS 5.2

Apple has released a new version of WebObjects, the company's suite of tools and object-oriented frameworks, that lets users develop and deploy scalable, reusable Web and Java applications. WebObjects 5.2 continues Apple's move into standards-based services adding more features for developers.

In addition to adding Web services to this version of WebObjects, Apple also made improvements in other areas including J2EE and Java Web Start integration.
www.apple.com

Oracle Releases Latest Version of Oracle Enterprise Manager

(*San Francisco*) – Oracle Enterprise Manager 4.0 provides the tools you need to manage every aspect of the expanded Oracle Infrastructure. It has a repository-backed Knowledge Center that collects metrics on availability, performance, and system configuration data. Enterprise Manager 4.0 also has a an HTML management console that assures IT managers top performance from their entire IT infrastructure – databases, Web and application servers, business applications, Web services, operating systems, hardware platforms, storage devices, and the network.
www.oracle.com

EVER-BOOMING BORLAND NOW ACQUIRES TOGETHERSOFT

by Jeremy Geelan

TOGETHERSOFT CORPORATION, the company that recently acquired WebGain, found out recently that it in turn will be acquired by Borland Software Corporation.

With this transaction, software giant Borland is notching up its sixth acquisition this year alone, and its measure of respect for the quality of TogetherSoft and its 4,000 customers is clearly reflected in what analysts have called the "very reasonable" acquisition price of $185 million – comprised of $82.5 million in cash and 9,050,000 shares of Borland common stock.

"It's been a very exciting couple of weeks," says Borland's senior VP of business development, Ted Shelton, speaking exclusively to *JDJ Industry Newsletter*.

The strategy behind the TogetherSoft move is interesting, as is the question of assessing the consequences it may or may not have on the Java tools market.

It doesn't sound, for example, as if there will be much need or houseroom in Borland for the WebGain suite of products that TogetherSoft just acquired. What's the point of even bothering to acquire the WebGain IDE, Visual Café Enterprise, along with the rest of the company?

Shelton does not duck the issue. "I'll concede the point that there are elements of TogetherSoft's product path – the acquisition of WebGain, for example," he says, "that overlap with the product path of Borland."

"We believe JBuilder is the de facto standard for Java development," he continues, not needing to spell out what that inevitably means for the destiny of WebGain Studio Java development tools. Borland is hardly likely to champion an alternative to JBuilder when it comes to automating the creation and integration of Web sites for online businesses.

In general though, Shelton contends, the two product lines are genuinely complementary rather than mutually exclusive. With TogetherSoft's products, customers get software and services they can use to integrate software design, deployment, and debugging processes. With Borland, they get technology used in the development, deployment, integration, and management of software applications.

What will happen to Together ControlCenter, now in version 6.0 and fast catching up with Rational Rose and Rational XDE as an integrated application development environment?

According to TogetherSoft founder Peter Coad, also speaking exclusively with *JDJ Industry Newsletter*, the future of ControlCenter software is in very safe hands indeed.

As he sees it, it will be possible to move beyond the TogetherSoft vision of model-driven design, toward a more integrated situation in which "it's not just the models and code that are always in sync but everything that's involved in the development process: programmers, designers, and managers."

• • •

To read more about the acquisition, go to www.sys-con.com/java/article.cfm?id=1733.

Dealing with Complexity

What Ajit Sagar describes in his editorial ("Complex Relationships" Vol. 7, issue 10) is exactly what keeps small development houses like us away from J2EE. We just don't have the resources to deal with the complexity. Servlets are okay; JSPs are OK; JMS maybe; EJB forget it. Excluding such a huge market that's made up of small developers is a major failing of this technology. Until it is addressed, it will continue to flounder and be susceptible to the likes of .NET. That's my two cents.

Joe Preston
drachma22@hotmail.com

I'm with you. This is a valid concern a lot of smaller vendors have. The good news is that a lot of cheaper yet sophisticated tools are becoming very popular in the J2EE and XP arena. And given the state of the current economy, companies are taking a serious look at these as alternatives for selecting vendors for building their applications. I believe that to develop an application using J2EE, you need a good IDE that offers a consolidated environment for developing, testing, and deploying J2EE applications. I work for a small company and our products are EJB-based, but I rely heavily on my IDE to keep a lot of things in synch.

Ajit Sagar
ajit@sys-con.com

Making the Move to Java

I first heard about Java in 1995 from a short article in *Business Week*. Not so odd, considering that most RPG programmers code for the AS/400, and have a considerable amount of business-process knowledge. By and large, we are average programmers who produce above-average work, given the excellent platform from IBM ("From COBOL or RPG to Java" by Dante Castiglione [Vol. 7, issue 10]).

Becoming a Java programmer requires not only learning Java, but how to use embedded SQL, what HTML is all about, TCP/IP concepts, hierarchical directories (we're spoiled with a flat library system on the AS/400), and other ancillary software technologies that are seldom used in the green-screen world.

The transition can be done, but you must have the passion to work through it. Join or start a user's group, and focus on doing JSPs at first to Web-enable the AS/400.

Doug Smith
dsmith@netroutines.com

Leave Nothing to Chance

Although I liked NTRU a lot and think their fast PK algorithms have a real future in the mobile space, it's still worth noting that NTRU algorithms are relatively young ("Unlimited Encryption on Limited Devices" by Bill Ray [Vol. 7, issue 11]). A few weaknesses have been found in the past year.

Any new crypto algorithm must go through peer review processes and fix all holes before it is robust. I hope NTRU will go through this quickly, but for users, security is leaving nothing to chance…

Michael Yuan
juntao_yuan@yahoo.com

JSuite 6.0

I liked Paul Frey's review of JSuite 6.0 (Vol. 7, issue 11), but what about the performance of the Infragistics components? This seems to be the issue with Swing, so I'm curious about these components?

Tim Prokuski
timsmail@yahoo.com

You're right, I didn't address performance. I approached this review from the standpoint of ease of use and integration in the software development cycle. JSuite 6 has some products based on Swing and some that aren't. I used the Swing-based components for their cross-platform ability. The products that are based on Swing suffer from the same performance issues that Swing has suffered from. That said, let me reiterate that I originally had been working with Java 1.2, and now with 1.4 I've noticed a vast improvement in Swing performance. But I also think that performance is very subjective, especially when it comes to GUI components.

Paul Frey
paulgfrey@yahoo.com

The 'Patterns' Revolution

I think Alan Williamson in his editorial "Design Pattern Snobs" (Vol. 7, issue 11) is mixing two issues into one rant. There are three effects of the "patterns" revolution.

The "Good" Patterns are one of the best things to happen to our industry. We're slowly moving toward a unified terminology. Now, architects can convey vision and ideas to software builders using well-known microarchitectures (yes, patterns).

With the "Bad" – algorithms, data structures, components, and other lingo-building revolutions – there is still not enough understanding on how to apply these wonderful patterns. A lot of people are working with patterns in a vacuum.

The "Ugly", of course, are the slew of people who read one or two articles and randomly throw pattern names into a conversation to feel better about themselves. These people are just devaluating the power of patterns, but as with any other good thing it's just part of the evolution of our trade.

Brian Sam-Bodden
bsbodden@integrallis.com

International

JDJ Edge 2003

CONNECTING THE ENTERPRISE WITH JAVA, WEB SERVICES, XML, AND .NET

March 18-20, 2003
Boston, MA

JDJ EDGE conference&expo

web services EDGE conference&expo

XML EDGE conference&expo

.NET EDGE conference&expo

Featured technologies and topics will include:

- Java Security
- JMS
- Enterprise Applications using Java
- TCP/IP
- JDO
- JXTA
- Relational database mapping
- Servelets

For more information visit
www.sys-con.com

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 Java Benefits, the industry pulse and proven strategies.

Contact information: U.S. Events: 201 802-3069 or e-mail grisha@sys-con.com • European & Asian Events: 011 44 208 232 1600

OWNED BY SYS-CON MEDIA
PRODUCED BY SYS-CON EVENTS

Boston call for papers opens November 11, 2002.

Submit your papers online at:
www.sys-con.com/webservices2003east

Rough Seas, Sinking Ships, and Lifeboats

Navigating the job market

WRITTEN BY
BILL BALOGLU & BILLY PALMIERI

As 2002 draws to a close, many of us find ourselves reflecting on the past year. There are many things we can be thankful for, primarily that this year wasn't nearly as cataclysmic as last year. Unless, of course, you happen to be one of those CEOs who was busted big time.

In many cases, the highest level of the corporate food chain is finally feeling the sting of the technology industry's "boom and crash." But those in the engineering trenches have been desperately treading water for the last couple of years.

An unfortunate result of this recent history is the following scenario that we've seen played out in numerous interviews and across many résumés. Perhaps the story may be familiar to you as well.

A senior engineer we know, who spent 10+ years going up a logical career ladder at a well-established company, was bitten by the pre-IPO bug in late 1999. He interviewed with a hot new startup, got a great offer, and gave notice to his long-time employer.

But on his first day at the new company, he discovered that the VP of engineering who hired him was gone. Six months later the company folded completely and the engineer set out on an odyssey from one job to another, only to find those companies downsizing or folding.

Suddenly the engineer's focus shifted from "Where is my career going?" to "Where is my next paycheck coming from?"

Through friends and networking contacts, he was able to get jobs unrelated to his core expertise, but he's smart, able to pick up new skills quickly – and there is that mortgage to consider. At first the move seemed interesting and exciting as he started learning new technologies. But the pattern of cutbacks and layoffs continued.

The result is a résumé that now lists three or four different jobs since 2000; they look like contracts, but were actually full-time positions. Anyone in a hiring position in the technology industry understands what's been going on over the past few years, and that companies folding is hardly a reflection on the candidate in front of them.

However, the biggest problem is that the multiple, short-term positions now listed on the résumé are in unrelated technology fields. With expertise that appears to be all over the map, it's increasingly difficult for him to get hired.

Especially now, as the industry is tightening up, managers are looking for candidates with more expertise in specific areas, not jacks of all trades.

This put the engineer in a twofold quandary with two burning questions: How do I get a job, or keep my current job? And how do I get my career back on track?

The biggest problem now is fear. Engineers don't want to leave their current job even if it's a dead end or not relevant to their primary skills. We see a lot of people who are staying on a sinking ship because they feel, "the devil they know is better than the devil they don't know."

Our best advice to engineers who need to steer their careers toward more reasonable shores is:

- Do everything you can to stay working in your area of expertise, even if it's for a company that few have heard of. If you've been building core platforms for product companies and are currently in a small startup doing the same, stay there. Don't move to a network company building applications because you think they have a better business model. A good business model today could be another bankruptcy tomorrow.
- Companies are looking for engineers who can come in and hit the ground running. Try to build on your core expertise and this does not mean simply technologies. Become a specialist in one area or domain. Companies are looking for engineers who can come in with proven experience in very specific areas. Right now companies are in the market to hire engineers with deep knowledge.
- Candidates often think having diverse experiences on their résumé paints a picture of a seasoned engineer with a multitude of talents. However, in these times it confuses managers and dilutes your strength. Keep your résumé focused. Make sure whoever reads it knows what you can do.
- Companies are in the mindset of swapping out engineers. They're strengthening their teams by removing the wheat from the chaff and adding senior engineers. You are valuable if you can bring specific talents to the company.
- Stay focused and don't panic. Things will turn around, and when they do, specialization will be even more in demand.

AUTHOR BIOS

Bill Baloglu is a principal at ObjectFocus (www.ObjectFocus.com), a Java staffing firm in Silicon Valley. Bill has extensive OO experience and has held software development and senior technical management positions at several Silicon Valley firms.

Billy Palmieri is a seasoned staffing industry executive and a principal at ObjectFocus. His prior postion was at Renaissance Worldwide, where he held several senior management positions in the firm's Silicon Valley operations.

jdjcolumn@objectfocus.com

JAVA DEVELOPER'S JOURNAL ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
AccelTree	www.acceltree.com		32
Altova	www.altova.com		27
AltoWeb	www.altoweb.com		39
Apple Computer, Inc.	www.apple.com/xserve		4-5
BEA	www.dev2dev.bea.com/useworkshop		9
Borland Software Corp.	www.borland.com		45
Bowstreet	www.bowstreet.com/easy		19
Crystal Decisions	www.crystaldecisions.com/cr9/002/	888-333-6007	31
Engenuity Technologies	www.jloox.com	800-361-6424	43
Evolution Hosting	www.evolutionhosting.com	866-257-4809	63
IBM	ibm.com/developerWorks/webservices/start		11
IBM	ibm.com/websphere/opentools		21
InetSoft Technology Corp.	www.inetsoft.com/jdj	888-216-2353	49
Infragistics, Inc.	www.infragistics.com	800-231-8588	14-15
iSavvix	community.isavvix.com	1-866-iSavvix	53
Jinfonet	www.jinfonet.com/jdj12.htm	301-838-5560	29
KeyLabs	www.keylabs.com	888-KayLabs	44
LinuxWorld Conference & Expo	www.linuxworldexpo.com		59
Macromedia	www.macromedia.com/go/jrun4jdj		33
Motorola	www.motorola.com/developers/wireless		6
/n software inc.	www.nsoftware.com		41
n-ary	www.javaSOS.com		61
Northwoods Software	www.nwoods.com/go/	800-434-9820	68
ObjectFocus Inc.	www.objectfocus.com	408-973-8200	77
Oracle Development Tools User Group	www.odtug.com		51
Our World Live	www.ourworldlive.com		Cover III
Parasoft Corporation	www.parasoft.com/jdj12	888-305-0041	37
Precise Software	www.precise.com/jdj	800-310-4777	35
QUALCOMM Incorporated	www.qualcomm.com/brew		55
Rational Software	www.rational.com/offer/roadmap4		13
Sitraka (now part of Quest Software)	www.sitraka.com/jclass/jdj	800-663-4723	23
Sitraka (now part of Quest Software)	www.sitraka.com/jprobe/jdj	800-663-4723	47
Sitraka (now part of Quest Software)	www.sitraka.com/performasure/jdj	800-663-4723	Cover IV
Sonic Software	www.sonicsoftware.com/jdj	800-989-3773	Cover II
SpiritSoft	www.spiritsoft.com/climber		25
SYS-CON Media	www.sys-con.com		69
SYS-CON Subscription Offer	www.sys-con.com/suboffer.cfm		74-75
WebAppCabaret	www.webappcabaret.com/jdj3.jsp		24
Web Services Edge 2003	www.sys-con.com	201-802-3069	79
Zero G	www.zerog.com	415-512-7771	3

Next Issue...
Java & Linux Focus

SEEING IS BELIEVING WITH JAVA3D

Java3D is no newcomer to the Java API world; however, it has suffered from slow acceptance due to the general resistance to client-side Java. Now that machines are faster, hardware 3D accelerators are a dime a dozen, and newer JVMs rival native code, client-side Java and 3D graphics are finally making headway. This article presents a brief background on Java3D, a few example applications, then finally some hints on how to use Java3D in the real world.

PACKAGING JAVA APPLICATIONS FOR OS X

Java on OS X is a first class citizen. You can integrate your app so well that users probably won't even know they're using a Java application. This article shows you how to create great-looking double-clickable applications while we wait for Apple to get the kinks out of their J2SE 1.4 port.

JAVA DATA OBJECT – WORKS TRANSPARENTLY

JDO is a standard API that's used generically to store, retrieve, and query user-written object classes to and from a data store. What makes JDO stand out among other persistence options is that it's easy to use and flexible. Learn the basics of JDO and how it can help you with your development.

PLUS

TOP 10 TIPS FOR JAVA DEVELOPERS WHO ARE DEVELOPING ON THE LINUX PLATFORM

GET THE INSIDE SCOOP ON THE BLACKDOWN PROJECT

IN-DEPTH REVIEWS OF PRODUCTS FOR THE LINUX PLATFORM

What Is OpenSymphony?

WRITTEN BY
JUSTEN STEPKA

OpenSymphony is a collection of Java open source projects that provides a foundation for building J2EE applications. Each application tends to build from another OpenSymphony component in a loosely coupled manner, providing a best-of-breed option.

The OpenSymphony initiative has been running strong for about three years now and has been in a state of production readiness from almost the beginning. Many developers use the OpenSymphony projects as an alternative to the Apache-Jakarta project due to its "if you build it they will come" attitude. OpenSymphony developers take a much different approach to their projects: add code that is fully unit tested and ready for production, as opposed to code that might fulfill a bullet-list requirement immediately.

One of the more popular OpenSymphony projects is SiteMesh. Imagine, for example, that you need to maintain a collection of pages for a particular Web site where each page shares the same look and feel in navigation. SiteMesh enables the developer to quickly apply a decorator to a specific page that would cascade the changes through the entire page set. This works by using the servlet filter API to analyze the HTTP request by mapping specific page requests to decorators that are defined in a configuration file. One of the more common uses of this would be for a header and footer.

OSCache is an API used to perform fine-grained dynamic caching of JSP content. Of course, you can use OSCache for much more, such as caching search results or another type of dynamic data on your model layer. The most powerful feature of the package is in the JSP taglib section. With the JSP taglib code, the developer has the ability to cache everything, from the whole page to a specific section! OSCache allows you to immediately hit the ground running with the easy-to-use API in any Java application.

> "WebWork does much of the same as Apache-Struts but provides a much cleaner and simpler approach"

WebWork does much of the same as Apache-Struts but provides a much cleaner and simpler approach. One of the key advantages of the framework is the decoupling of the controller from the traditional view technologies such as JSP, Velocity, and XML/XSLT. WebWork can easily be plugged into a standard Java application, SOAP, or testing suite, such as JUnit. The modularization of the project makes WebWork an easy choice when deciding on a controller for building a J2EE application.

OpenSymphony is not without its faults, though. The documentation and examples for the WebWork and OSWorkFlow projects are more than lacking. However, the community and teams have recognized their weaknesses and are actively working on providing better documentation and examples for upcoming releases.

Don't let the lack of documentation hold you back, though. All the projects are extremely valuable and integrate with each other flawlessly at a production-quality level that other open source projects should follow. The upfront investment time to learn OpenSymphony will pay for itself many times over and will leave you wondering why you didn't start using the projects sooner.

Reference

- *OpenSymphony Project:* www.opensymphony.com

AUTHOR BIO
Justen Stepka is an enterprise system architect for digital @ JWT Minneapolis. He develops Web-based applications for Fortune 500 companies and is actively involved in the design and research of emerging Web technologies.

jstepka@hoderi.com

Java Dudes

Why is your GUI square?

Develop advanced GUIs and reduce development time with out-of-the-box APIs for true free form shapes, transparencies and dragability

HIP™ PJ HIP™ J2

Too HIP™ to be square

HIP™ is the advanced user interface framework that transcends the limitations of AWT & Swing and revolutionizes the development of Java™ GUIs in 100% Pure Java™.

Broad platform compatibility

HIP™ 1.0 PJ is fully compliant with PersonalJava™ version 1.2 or newer

HIP™ 1.0 J2 is fully compliant with the Java™ 2 Standard and Enterprise Editions

☛ Download a full version for free evaluation at www.ourworldlive.com

BUY NOW – get 50% off

Buy HIP™ until November 30, 2002 and get 50% off the standard Developer License price.

OurWorldLive
Effortless Information Access

Get beyond metrics.

sitraka PerformaSure™

Reach your J2EE performance goals.

Metrics are useful. In fact, theyre often the first step in uncovering a performance problem in your J2EE systems. But lets face it, its not enough to detect the symptoms of a problem. You still need to solve it. Thats why Sitraka PerformaSure correlates all metrics with individual transaction response times. It s the difference between performance intelligence and just plain data.

Our exclusive Tag and Follow technology traces the end-to-end execution path of transactions through each component and server of a distributed J2EE system, with method-level timing captured every step of the way. Hotspots are intuitively color-coded for further drill-down. The result? Faster, more accurate performance analysis so you can reach your performance goals sooner.

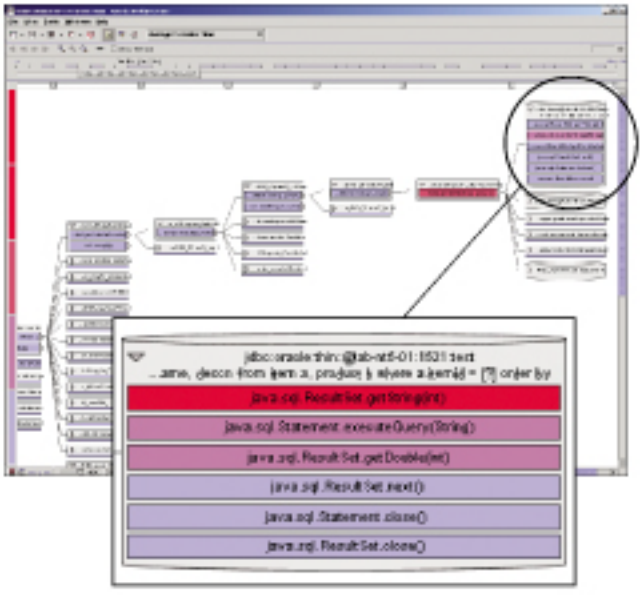
Track down J2EE performance problems including:

¥inefficient and excessive database calls

¥inefficient servlets, JSPs, EJBs, and Java classes

¥network misuse including frequent/large object serialization

¥application server settings that limit scalability
(EJB caching, JDBC pool sizes, etc.)



The Transaction Tree provides an end-to-end method-level view of any transaction s path through the distributed J2EE system. Clearly see component interactions within the application architecture on a per-transaction basis, with performance hotspots highlighted in red.

Take a walkthrough of PerformaSure today at:
www.sitraka.com/performasure/jdj

sitraka
the Java™ advantage