ORACLE DELIVERS NEW RELEASE: **ORACLE9*i* JDEVELOPER** P.76

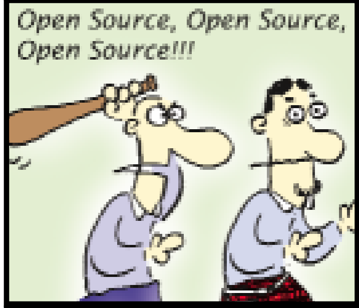
JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

November 2002 Volume:7 Issue:11

JAVADEVELOPERSJOURNAL.COM

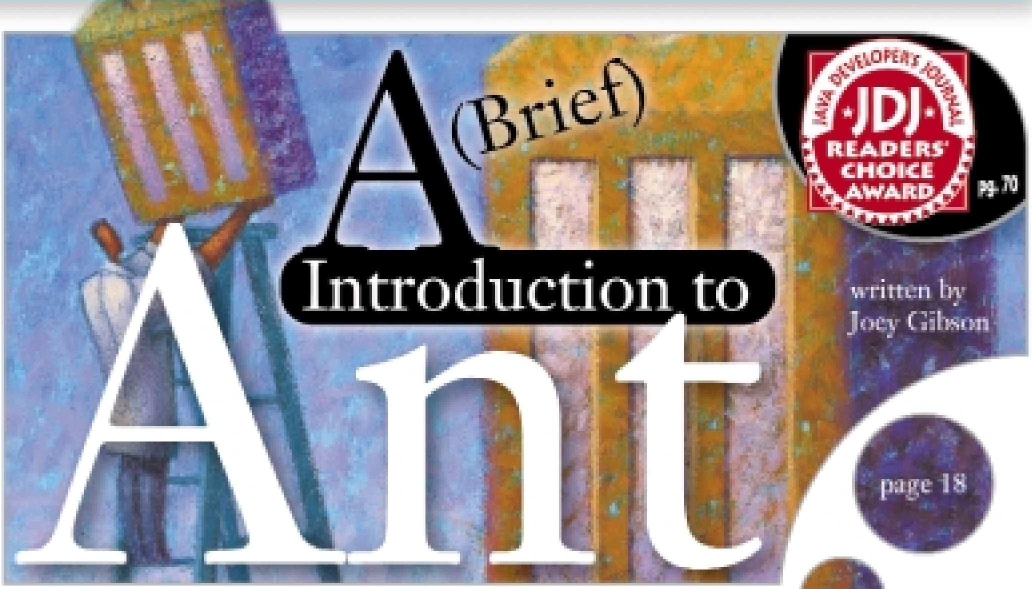
JDJ's Java & Linux Focus Issue on Newsstands in February!



RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2003

$5.99US $6.99CAN

SYS-CON MEDIA



A (Brief) Introduction to Ant
written by Joey Gibson
page 18
JDJ READERS' CHOICE AWARD pg. 70

sonic

www.sonic.com

zero g

www.zerog.com

apple

www.apple.com

apple

www.apple.com

motorola

www.motorola.com

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)*, • RICK ROSS *(President, JavaLobby)*, • BILL ROTH *(Group Product Manager, Sun Microsystems)*, • BILL WILLETT *(CEO, Programmer's Paradise)* • BLAIR WYMAN *(Chief Software Architect IBM Rochester)*

EDITORIAL

EDITOR-IN-CHIEF: ALAN WILLIAMSON
EDITORIAL DIRECTOR: JEREMY GEELAN
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, STEVEN BERKOWITZ, JASON R. BRIGGS, BRECK CARTER, PAUL FREY, JEREMY GEELAN, JOEY GIBSON, ENRIQUE PÉREZ GIL, VISHAL GOENKA, TOM LARAMEE, JOSEPH OTTINGER, BILLY PALMIERI, BILL RAY, ROGER RITTER, AJIT SAGAR, ALAN WILLIAMSON, BLAIR WYMAN

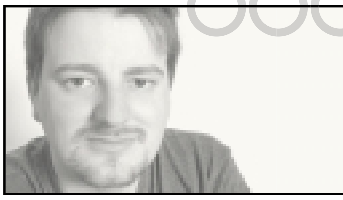
SUBSCRIPTIONS:

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

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

EDITORIAL OFFICES:

SYS-CON MEDIA 135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9600
JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is published monthly (12 times a year) for $49.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. POSTMASTER: Send address changes to: JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.



ALAN WILLIAMSON EDITOR-IN-CHIEF

Design Pattern Snobs

Have you noticed lately how the word *pattern* seems to be creeping into general musings and dialogue more and more? Like name-dropping, it's consciously woven into the fabric of the conversation as a way to assert a certain level of understanding and credibility. Mention the latest design pattern and suddenly your peers will see you as a genius of software engineering, "…you see I have employed the Decorator pattern for this particular class…" While you're fighting the urge to give them a good slap, allow me to let you into the big secret. There is none!

When perusing various article submissions for our beloved magazine, I am constantly surprised to see how laden down with buzzwords the proposals are, as if vainly attempting to show off their superior knowledge by dazzling our editors with fancy acronyms and big words. I had an exchange with one potential author, who I would class as the typical design-pattern snob. I'm sure you've come across them at some point. They're the ones who love using this terminology, and chastise anyone who doesn't understand or uses an unofficial pattern.

To that end I had to do something – I had reached the saturation point. Although I am a young fellow, I have to hold my hand up and say I was educated in an era before design patterns were even invented. Yes, it's true, such a time did exist. In my day they were called *data structures*. Now admittedly, this doesn't sound half as sexy as design patterns, but they were the exact same thing without today's buzzwords.

To all those developers who are getting frustrated with the constant use of the word *patterns*, and who feel they may be missing out on some great new movement, I say to you: don't panic. They're just repackaging the stuff you've been doing for years in a format that can be openly discussed. For example, the class you designed that can have only one instance created – in the pattern world, this would be known as a Singleton class. Easy, eh? Not shrouded in as much mystery now. A design pattern does not specify any code, nor does it even specify a template, merely a way of doing something. A pattern!

The next time you get into a discussion on design patterns, be it at an interview or over the water cooler, stay calm, and merely ask the offender to explain what exactly he or she means. I would guess that 9 times out of 10 you've used it many times and didn't even realize you were a design pattern follower!

• • •

Last month I highlighted how delighted I was with a great client-side technology, www.thinlet.com, and how it was a wonderful example of how Java, in the right hands, can really rock. This month I have found another. It's not a tool for Java developers, however; it's more for the masses. Check out www.freedomaudio.com, a lightweight Java audio tool for listening to streaming MP3 and Ogg files inside your browser. It works very well, and Kendal, the man behind it, has put a lot of effort into writing his own libraries to ensure that they're small, as opposed to using some of the bloated official Java APIs. Yet another shining example in which small can be beautiful, and creativity and engineering can prove that Java is still the only one to win against Flash and other plug-in technologies.

Keep up the good work, people. We are proud of you. Your community needs you.

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.

Sun Is Losing Its Way

WRITTEN BY JOSEPH OTTINGER

I've been actively involved with Java development in one way or another since 1996, including working with some of the original issues of the servlet specification, the early adaptation of the EJB spec, and migration to JSP not long after it became an official part of the J2EE spec. I remember when Rick Ross first sent his e-mails for Javalobby on Usenet; I remember playing with the specs to discover if the grail was to be found in them as promised.

It hasn't come through for me. That's okay, because technology is and will always be a moving target; I've refined what the grail is as I've come near it many times, and I will continue to do so as long as my career is not completely stagnant. What worries me, though, is the thought that the grail is actually moving further away as time goes by, instead of staying just out of reach, like a carrot for a hungry dray horse.

The Java Community Process was supposed to give relevant people in the Java Community a chance to influence the specifications in positive ways. It's guaranteed to be an arduous process, and it has lived up to its billing. What Sun needed to do was guide the expert groups down a path laden with thorns to determine not only what was best, but how to get what was best implemented. What it did was let the expert groups descend into a morass where those who shouted most were heard.

The Enterprise JavaBean specification is a good example of this. EJB 1.0 fell critically short of being easily deployable; EJB 1.1 added some great features, and 2.0, while fulfilling some of the void left by the 1.1 specification, has sadly started to invalidate EJB's promise. The best things about EJB remain the same; the new additions, such as local interfaces, simply add complexity where it's not needed. And that's where Sun's community process has failed. Nearly every container already had an implementation of local interfaces before they were added to the spec; they were added as a sop to a few major players who simply hadn't kept up with the "smaller" containers. Obviously, it was possible to do without a lot of overhead, yet Sun felt it was all right to make not only every container but every EJB coder work with extra interfaces as well as a separate use model. This is the JCP breaking down; instead of making life simpler, it made it harder and didn't actually finalize the most crucial areas left in the entity bean section of the specification, such as finalizing the managed relationships.

The 1.4 API's logging model is another example of this: taken nearly verbatim from Log4J, it's a floodgate-based model, where each category can be set up individually, and as messages rise above a deployer-specified "message level," they become visible in the output stream. This is nice, certainly nicer than System.out.println() strewn throughout code, but the introduction into the 1.4 API would have been the perfect time to present something more powerful by default, such as a bitmasked logger, where you would be able to say you wanted to see all "startup," "info," and "fatal" events in a given category, and in another you wanted to see only "startup" and "shutdown" events. Instead, Sun continued to take the easy way out, repackaging Log4J according to various contributors in the expert group and letting it go.

I would have liked to see Sun actually leverage the expertise of the Java community – it could spend time developing the language and application interfaces while accepting the contributions of programmers who could improve the core implementations. That would yield massive benefits: a best-of-breed would appear in the language API and application developers would feel empowered and more invested in Java. A best-of-breed would probably have prevented things such as local interfaces (which actually do have a beneficial performance impact, just not enough of one to justify their cost), and allowed a more open discussion of other relevant APIs.

As Sun continues to allow the JCP to run wild, contributing APIs that don't actually improve the usage patterns of the language, Java will grow more and more irrelevant, until the weight of the poor APIs drags it into oblivion. Sun needs to listen to its own best practices groups, and eliminate the cruft introduced by users who very competently implement things that run counter to the mindset that made Java the useful tool it is today.

joeo@enigmastation.com

AUTHOR BIO
Joseph Ottinger is a consultant with Fusion Alliance (http://fusionalliance.com) in Indianapolis, and is one of the administrators of and contributors to the OpenSymphony project (www.opensymphony.com).

SYS-CON MEDIA

PRESIDENT AND CEO
FUAT A. KIRCAALI fuat@sys-con.com
COO/CFO
MARK HARABEDIAN mark@sys-con.com
VICE PRESIDENT, BUSINESS DEVELOPMENT
GRISHA DAVIDA grisha@sys-con.com

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

EDITORIAL
EXECUTIVE EDITOR
NANCY VALENTINE nancy@sys-con.com
ASSOCIATE EDITORS
JAMIE MATUSOW jamie@sys-con.com
GAIL SCHULTZ gail@sys-con.com
JEAN CASSIDY jean@sys-con.com
ASSISTANT EDITOR
JENNIFER STILLEY jennifier@sys-con.com
ONLINE EDITOR
LIN GOETZ lin@sys-con.com

PRODUCTION
PRODUCTION CONSULTANT
JIM MORGAN jim@sys-con.com
LEAD DESIGNER
LOUIS F. CUFFARI louis@sys-con.com
ART DIRECTOR
ALEX BOTERO alex@sys-con.com
ASSOCIATE ART DIRECTOR
RICHARD SILVERBERG richards@sys-con.com
ASSISTANT ART DIRECTOR
TAMI BEATTY tami@sys-con.com

WEB SERVICES
VICE PRESIDENT, INFORMATION SYSTEMS
ROBERT DIAMOND robert@sys-con.com
WEB DESIGNERS
STEPHEN KILMURRAY stephen@sys-con.com
CHRISTOPHER CROCE chris@sys-con.com

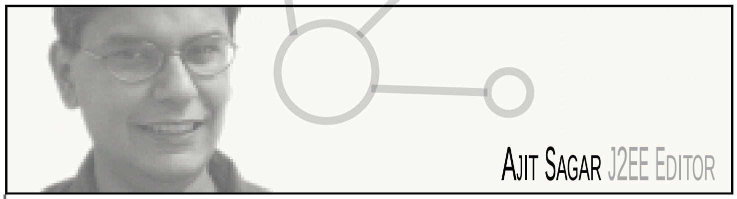
ACCOUNTING
ASSISTANT CONTROLLER
JUDITH CALNAN judith@sys-con.com
ACCOUNTS RECEIVABLE/COLLECTIONS SUPERVISOR
KERRI VON ACHEN kerri@sys-con.com
ACCOUNTS PAYABLE
JOAN LAROSE joan@sys-con.com
ACCOUNTING CLERK
BETTY WHITE betty@sys-con.com

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

CUSTOMER RELATIONS
CUSTOMER SERVICE REPRESENTATIVE
MARGIE DOWNS margie@sys-con.com
JDJ STORE MANAGER
RACHEL McGOURAN rachel@sys-con.com

bea

www.bea.com



AJIT SAGAR J2EE EDITOR

eXtreme J2EE

When I first started programming, it was with a small company. Life was simple. I understood all the requirements, and knew all the aspects of the application and how to pull everything together. If I was working with a team of programmers, the projects were small enough that the team knew each other's code thoroughly. Mi code es su code. Software engineering was easy to follow; code and design walk-throughs involved everyone, and, given time, any person could pretty much handle the whole project on his or her own.

I moved on in my career and, in the last decade, as technology started to grow at a phenomenal rate, the process of "pulling it all together" became far more complex and had a lot more nuances than a single person or team could handle. Distributed computing, by its very definition, requires distributed components to work together as an integrated application. This introduces the need to independently test and deploy the components, to provide an appropriate level of abstraction, and then to test the application as a whole.

The J2EE platform's purpose in life is to facilitate the development and integration of distributed components. Sun provides the APIs and the framework. However, the actual tools for testing and deploying an application are provided by IDE tools vendors. Fortunately, J2EE IDEs have come a long way in offering features that support the design and testing as well as deployment of J2EE-based applications.

As distributed programming platforms have evolved over the last decade, so too have the design methodologies. About four years ago, you got the J2EE APIs (which were not yet "J2EE") from Sun, the J2EE implementation and container for distributed components from your app server vendor of choice, and the development environment from your favorite IDE. To coordinate between development, testing, and deployment, you had to manually make these different environments work together.

Over the past few years, several things have happened to make the development of J2EE applications both easier and harder. It's harder because the complexity of APIs has grown to an extent that the developers can concentrate on only a part of the big picture. The architect who designs the solution has the big picture in mind, but had to leave the details of subsystem design, testing, and deployment to the individual subsystem designers. Development of J2EE-based applications has become easier because the tools have become more integrated and alternative, less heavyweight methodologies have become very popular.

eXtreme programming has gained a lot of popularity in the J2EE universe overall because it offers a practical and efficient way to manage large or small projects without getting bogged down in an overly complex design and an unmaintainable development cycle. And XP has become very popular in the Java environment because it addresses the needs of distributed J2EE application development in a palatable way. IDE and app server vendors have added support for J2EE tools using XP as well as support that enables teams to develop software based on XP principles. The leading app server and IDE vendors typically include support for tools, such as Ant for building and deploying software and JUnit for testing, and editing support for refactoring. The XP principles apply well to a distributed development environment by reinforcing simple design and continuous integration, refactoring coding standards, and code sharing through pair programming. While these principles apply well to all projects, the supporting Java tools make the application of these principles pragmatic.

It would be nice to have a tool that enforces pair programming. Maybe that's coming soon. Speaking of eXtreme J2EE, there are a couple of good books I'd like to recommend on this subject: *Java Tools for eXtreme Programming* by Richard Hightower and Nicholas Lesiecki (Wiley), and *Java Development with Ant* by Erik Hatcher and Steve Loughran (Manning Publications).

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.

J 2 E E I N D E X

ibm

www.ibm.com

JDJ Asks . . . Microsoft on .NET

Interview with Microsoft

Microsoft has been making major plays in the industry regarding Java with their .NET strategies, and a lot of disinformation has been floating around. Microsoft granted *JDJ* access to ask questions, offering you, the reader, the opportunity to find out, straight from the horse's mouth, what's going on in Redmond.

Concerned readers of *JDJ* posed the following questions.

<rob diana>:**What are the core differences in the .NET J# implementation and normal Java implementations?**
<microsoft>: Visual J# .NET is a tool for Java developers to use in building applications and services on the .NET Framework. It integrates the Java-language syntax into the Visual Studio .NET shell, and enables it to integrate with the 20-plus programming languages supported by the .NET Framework. It's for Java developers who want to build XML Web services or have an investment in Visual J++. First, we are implementing sufficient functionality to ensure that our existing Visual J++ customers will be able to take advantage of the .NET Framework with the smallest possible disruption. In future releases, we'll be examining development and usage patterns among our developers and offer additional support as appropriate.

<mica cooper>: **I use Java almost exclusively. Quite often the best solution for a Windows customer is to use Microsoft tools such as the one that makes a Java class an NT service. Are there any plans to update these tools? Are they in .NET or *not*? If they are there, do I have to use .NET Java or can I use standard Java?**
<microsoft>: The Microsoft SDK for Java provides the com.ms.service namespace, the Service base class, and the Jntsvc command-line tool. Developers can author Windows Services by extending the Service base class and using the jntsync tool to install the Service. The .NET Framework provides equivalent functionality via the System.ServiceProcess.Service class and InstallUtil command-line tool. As with any other language that supports the CLR, Visual J# .NET can be used to build a Windows Service that extends this new base class.

<richard hart>: **What is your respect/intentions for the Java language? You don't seem to view it as a standard, otherwise we wouldn't have the IE/Netscape issue with running applets. Are you looking to support Java 1.4 ever? Or will you be taking Java off into a different direction completely?**
<microsoft>: Our intention is to enable Java-language developers to take advantage of the Microsoft .NET Framework. If you're a developer who is comfortable using the Java-language syntax and wish to start building third-generation applications on a true XML Web services platform, then Visual J# .NET will appeal to you.

<concerned java developer>: **Can you tell us the main factor that prevented you from working with Sun all those years ago? Why are you stuck with 1.1.3 of the specification?**
<microsoft>: At its core, our primary issue with Sun was whether Microsoft could innovate in Java. We wanted to make it possible for developers to build great Windows applications using the Java language and to leverage the full power, functionality, and integration of the Windows platform while also providing support for developers who chose to write cross-platform Java applications. Sun sued us to prevent that innovation. The settlement we reached with Sun only allows us to provide a JDK 1.1.4-era virtual machine, and our right to do that effectively expires in January 2004 when our ability to repair security vulnerabilities ends.

<jay miller>: **What has happened to J#? It was slated for Q2 release, and then Microsoft got *real quiet* about it.**
<microsoft>: Visual J# .NET was announced for public availability on July 1, 2002, at Tech-Ed Barcelona. You can download the released product from http://msdn.microsoft.com/vjsharp. We're very happy with the level of interest in Visual J# .NET and are already planning for future releases.

<rupal majmudar>: **Microsoft supports J# on the desktop with the VJ++ add-on to Visual Studio. Do they plan to support J# on Compact.NET/WinCE .NET platforms as well? If yes, when? If not, will they be saying so officially, or will they leave us guessing?**
<microsoft>: Currently the .NET Compact Framework doesn't have IDE/compiler support for J#. We are working closely with our customers and partners to determine the correct offering for Visual J# .NET in this space. We'll keep you informed.

<james ashton>: **Does Microsoft have any plans to upgrade the Java support in J# to anything approaching what we're all actually using (1.3.x/1.4.x)? If not, would there ever be a mechanism to allow a developer to specify a JDK to use as a base?**
<microsoft>: We're not building an implementation of Sun's proprietary Java platform. Our goal is to provide a development tool for building applications and services on the .NET Framework using the Java-language syntax. So developers who are interested in writing applications in Java for the .NET Framework will find a familiar syntax structure and core set of runtime libraries.

<franklin nwankwo>: **Why is Microsoft hesitant to incorporate adequate support for Java in the .NET Framework? Do they realize that Java developers will not JUMP that easily? What is needed is support for pure unadulterated Java in .NET as all other attempts at J# JUMP, et al, are doomed to spectacular failure.**
<microsoft>: We feel we have implemented a better platform for creating XML Web services – the .NET Framework. Once developers realize there's something out there superior to Sun's proprietary Java platform, as many have already, we feel that they will want to begin development on the .NET Framework. The JUMP strategy is designed to help move developer skills and existing applications to the .NET Framework.

<ken barber>: **Would it be fair or unfair to say that due to choices made by MS when designing the Win32 architecture long ago, Windows is built on a very dangerous foundation (ActiveX, for example) that *can't* be made secure, and which is likely to render .NET equally insecure?**
<microsoft>: No, it would not be fair to say that because it is fundamentally incorrect. We would recommend that you look at the many articles on the robust security features of the .NET Framework at http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=2800136910.

<ramakrishna kuppa>: **What's Microsoft's take on the observation and comments made by Java evangelist Rima Patel at http://java.sun.com/features/2002/07/rimapatel.html, related to adopting the MS .NET Framework. Through Rima, I heard the Sun story, and through MS, I'd like to hear a detailed reply to the same.**
<microsoft>: While the most charitable thing we can say about Ms. Patel is that she is woefully uninformed, we think it's

rational

www.rational.com

infragistics

www.infragistic.com

infragistics

www.infragistic.com

better to provide customers with the information to make their own decisions rather than engaging in name calling. The best source for information about how the .NET Framework compares is www.gotdotnet.com/compare. There customers will be able to find independent analysis of benchmarking claims as well as the code to actually replicate the systems used in testing (which we recommend) and articles on comparisons. In addition, at sites like www.gotdotnet.com, www.asp.net, and http://msdn.microsoft.com, you'll get a sense of the size of the .NET Framework developer community and how those developers are taking advantage of the new features and functionality of Microsoft's new platform for XML Web services.

<anonymous>: **Given that MS has the industry really excited about .NET in general, why do you think .NET has had so few results so far in terms of creating practical Web services?**
<microsoft>: Customers have been very successful with creating practical XML Web services. You can see lots of information about how customers are using the new platform at http://msdn.microsoft.com/vstudio/productinfo/casestudies/default.asp. Microsoft believes the adoption of XML Web services will come in waves. This first wave is about early adopters and pilot projects within the firewall. Customers will see how others are succeeding with such projects and will recognize similarities within their organizations where similar XML Web services will benefit them. As issues around security and reliability are further addressed by working with the industry on standards, such as WS Security, Microsoft believes that XML Web services will take off further.

<jon strayer>: **What are your plans for porting .NET to other operating systems (HP-UX, Solaris, AIX, Zos)?**
<microsoft>: We do have a noncommercial implementation of the C# and Common Language Infrastructure ECMA standards running on FreeBSD and on Windows XP, but customers shouldn't read anything into that except that those standards are true, open, platform-neutral standards from a recognized standards body. That said, we haven't announced any plans to implement the .NET Framework on other operating systems.

<anonymous>: **When will .NET support XUL (XML User Interface Language)?**
<microsoft>: We currently don't have any plans to support it.

<tom watts>: **I do a lot of work with servlets, and I believe the direct comparative component within the .NET Framework for this is ASP.NET. What are the major differences between ASP.NET and servlets?**
<microsoft>: ASP.NET provides two separate levels of infrastructure for supporting Web applications: the HttpRuntime and Page Framework. These are roughly analogous to servlets and JSP in J2EE. The HttpRuntime is a highly extensible framework to enable basic processing of Web requests. Developers taking advantage of the HttpRuntime develop classes that implement the IHttpHandler interface. This handler implementation is then registered to process a given request type (based on URI or extension). When a request matching the handler is received, the HttpRuntime will create an instance of the handler and call its IHttpHandler::ProcessRequest method. The Page Framework rests upon the HttpRuntime (in fact, the base class of all ASP.Net Web Forms implements IHttpHandler). This framework provides a RAD infrastructure for developing a Web application. In short, the Page Framework provides "VB for the Web" functionality, hiding the details of the HttpRuntime, making it faster and easier to develop Web applications.

<lee graba>: **Can the entire .NET Framework (or at least C#, the CLR, and the .NET libraries) be clean-room duplicated by any third party without any IP restrictions imposed by MS? Do you have patents on technologies within .NET, and, if so, will you renounce any intention to limit the work of the Mono project and .NET clones through patent enforcement?**
<microsoft>: The .NET Framework includes valuable intellectual property belonging to Microsoft. Just like any other company, we will review any action that may affect our intellectual property rights before making a decision about how to proceed. Any other speculation is just that, speculation.

<fred grott>: **Given MS's seven-year history of claiming that Java is a virus, I can see where the misinformation may be coming from. Is MS going to change this policy of FUD so that we may get open disclosure of the benefits of .NET and Java?**
<microsoft>: Microsoft has never claimed Java is a virus, but we're more than happy to have a discussion about the relative merits of competing technologies. It's very clear that once developers use Visual Studio .NET and the .NET Framework, they agree with us and the analysts that the .NET Framework is the leading platform for XML Web services.

<jeff duska>: **Why didn't Microsoft base .NET around Visual Basic instead of C#? Visual Basic already was using a P-Code VM. It just seems as if you like Java, but since you could/would not do Java you created C#. As a VB developer, I know this was my main reason for leaving the fold for Java development. When Microsoft says it's more important to clone Java than to support VB developers, I felt it was time for a change.**
<microsoft>: Actually, we based Microsoft .NET around the best of Visual Basic, ASP, and MFC/ATL . The existing Visual Basic runtime and p-code engine did not allow us to do things like inheritance and threading, which we felt were absolute requirements of a modern programming environment. Obviously, we're deeply committed to Visual Basic; it's the most widely used programming language. We modernized it; we removed the "glass ceiling" by providing full access to the platform; and we are ensuring that VB will continue to be the most productive programming environment and language moving forward.

<tom jordan>: **Platform independence is a Java feature taken for granted by Java programmers. On what platforms is the CLR currently ported?**
<microsoft>: Please see our answer to Jon Strayer.

<kunal>: **In the wireless space, how does Microsoft plan to address the features and popularity of Sun's J2ME that are missing in .NET? Are there plans to make .NET's Compact Framework more carrier-centric (like BREW, for example) or more application-centric, like J2ME?**
<microsoft>: Carriers and applications are not opposing goals. A strong link between the two is essential if the industry is to be successful in delivering innovative scenarios for users. We have a very attractive offering that enables the full wireless ecosystem. We enable the broadest number of developers to build compelling applications for mobile devices using their existing skills and knowledge. We enable devices to participate in the Web services ecosystem with technologies like .NET Compact Framework for smart client applications, and ASP.NET Mobile controls for Web

–continued on page 24

bowstreet

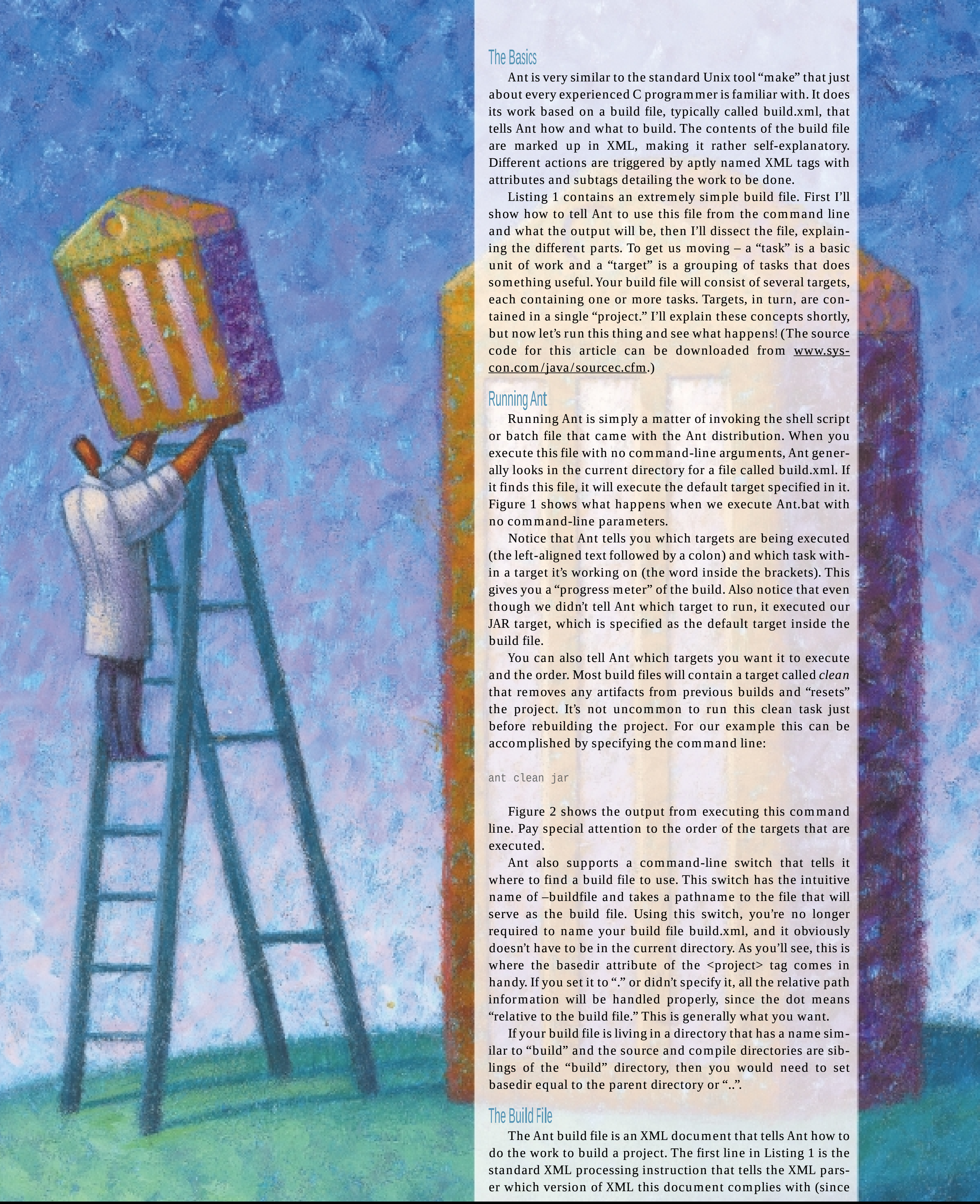
www.bowstreet.com

A (Brief) Introduction to Ant

Using a build tool to solve business problems

written by Joey Gibson

If you haven't heard of Ant before, it's an extremely useful Java build tool from the Apache Jakarta group that makes building Java projects a breeze. This article provides a short introduction to Ant so you can start using it in your own projects. I'll assume that you don't know Ant; however, a passing knowledge of XML is required since the format of an Ant build file is XML.



The Basics

Ant is very similar to the standard Unix tool "make" that just about every experienced C programmer is familiar with. It does its work based on a build file, typically called build.xml, that tells Ant how and what to build. The contents of the build file are marked up in XML, making it rather self-explanatory. Different actions are triggered by aptly named XML tags with attributes and subtags detailing the work to be done.

Listing 1 contains an extremely simple build file. First I'll show how to tell Ant to use this file from the command line and what the output will be, then I'll dissect the file, explaining the different parts. To get us moving – a "task" is a basic unit of work and a "target" is a grouping of tasks that does something useful. Your build file will consist of several targets, each containing one or more tasks. Targets, in turn, are contained in a single "project." I'll explain these concepts shortly, but now let's run this thing and see what happens! (The source code for this article can be downloaded from www.sys-con.com/java/sourcec.cfm.)

Running Ant

Running Ant is simply a matter of invoking the shell script or batch file that came with the Ant distribution. When you execute this file with no command-line arguments, Ant generally looks in the current directory for a file called build.xml. If it finds this file, it will execute the default target specified in it. Figure 1 shows what happens when we execute Ant.bat with no command-line parameters.

Notice that Ant tells you which targets are being executed (the left-aligned text followed by a colon) and which task within a target it's working on (the word inside the brackets). This gives you a "progress meter" of the build. Also notice that even though we didn't tell Ant which target to run, it executed our JAR target, which is specified as the default target inside the build file.

You can also tell Ant which targets you want it to execute and the order. Most build files will contain a target called *clean* that removes any artifacts from previous builds and "resets" the project. It's not uncommon to run this clean task just before rebuilding the project. For our example this can be accomplished by specifying the command line:

```
ant clean jar
```

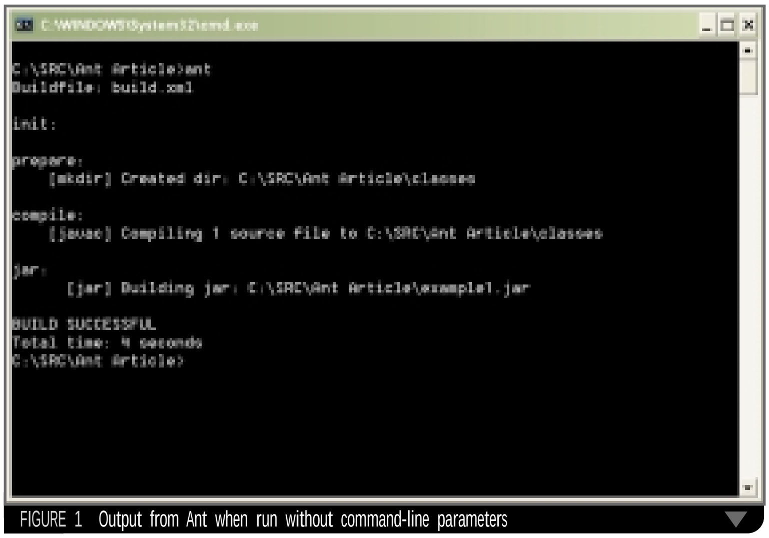
Figure 2 shows the output from executing this command line. Pay special attention to the order of the targets that are executed.

Ant also supports a command-line switch that tells it where to find a build file to use. This switch has the intuitive name of –buildfile and takes a pathname to the file that will serve as the build file. Using this switch, you're no longer required to name your build file build.xml, and it obviously doesn't have to be in the current directory. As you'll see, this is where the basedir attribute of the <project> tag comes in handy. If you set it to "." or didn't specify it, all the relative path information will be handled properly, since the dot means "relative to the build file." This is generally what you want.

If your build file is living in a directory that has a name similar to "build" and the source and compile directories are siblings of the "build" directory, then you would need to set basedir equal to the parent directory or "..".

The Build File

The Ant build file is an XML document that tells Ant how to do the work to build a project. The first line in Listing 1 is the standard XML processing instruction that tells the XML parser which version of XML this document complies with (since



```
C:\SRC\Ant Article>ant
Buildfile: build.xml

init:

prepare:
    [mkdir] Created dir: C:\SRC\Ant Article\classes

compile:
    [javac] Compiling 1 source file to C:\SRC\Ant Article\classes

jar:
      [jar] Building jar: C:\SRC\Ant Article\example1.jar

BUILD SUCCESSFUL
Total time: 4 seconds
C:\SRC\Ant Article>
```

FIGURE 1 Output from Ant when run without command-line parameters

there's currently only one version of XML, this line will always be set to version 1.0).

The <project> Tag

Line 3 is where the interesting content begins. The <project> tag is the only top-level tag that is allowed in a build file. In this example the <project> tag defines a logical name for this project (the name attribute) and specifies a default target (the default attribute) and a base directory for relative path specifications (the basedir attribute).

The name is not really used anywhere with an out-of-the-box Ant installation, but integrated development environments that understand and integrate with Ant will use the name to differentiate between different projects. Examples of Ant-aware IDEs include Eclipse, jEdit, and IntelliJ IDEA (URLs are provided at the end of the article).

The default attribute tells Ant which target to execute if you don't specify one on the command line. In general, the default target should be set to the one that's most useful or the one you execute the most; this will save some typing later on. In our case we want everything to execute, so we'll set the JAR target as the default.

The basedir attribute is used when building up relative pathnames. There are many properties set in a typical build file that specify directories or files. It's much easier and portable to specify these in a relative rather than absolute fashion. In other words, using the basedir attribute will allow you to pick up and move your entire build directory structure to a different disk or machine, without radically changing your build file. This is a big win, especially in a multideveloper scenario where different developers may not have the project directory in the same place on their machines.

Setting Properties

Just as with make, it's useful and indeed encouraged to set constants for often-used information to avoid retyping and copying. This is accomplished with the <property> tag that can appear either by itself directly inside the <project> tag or inside a <target> tag. The <property> tag has several options but we'll consider only one here: file. Supplying a file name (which will be found relative to the basedir attribute if no path is given) will create properties for each line in the file. The following is the properties file that we're using for this project.

```
1  projectname=example1
2  src.dir=src
3  jar.name=${projectname}.jar
4  build.dir=build
```

This is standard Java properties syntax. We are defining four properties, one of which is based on one other; the jar.name property on line 3 is based on the projectname property on line 1. The ${} syntax is Ant's way of dereferencing a property, substituting the property's value. You'll notice that on lines 9, 13, 14, 18, 19, 24, and 25 of the build file we use this same syntax – the same dereferencing syntax that works in the properties file works in the build file.

Note that we could have specified each of these properties directly in the build file, but it is much cleaner to move them out to a properties file. This makes the build file less cluttered and the properties a little easier to read.

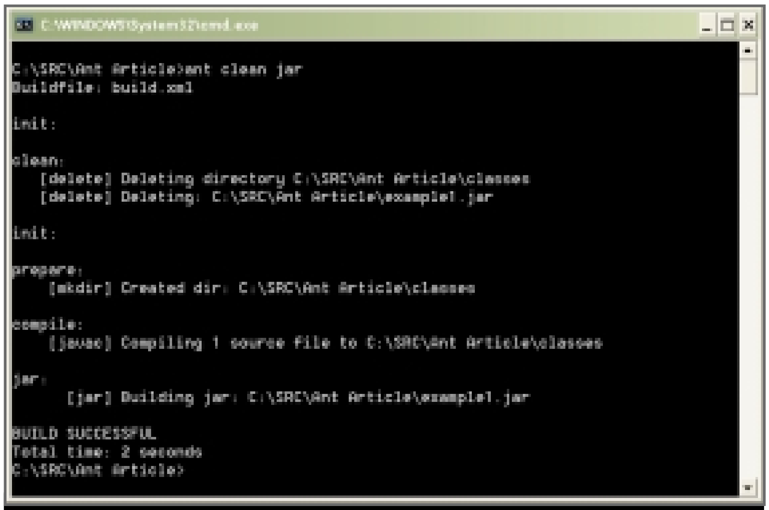
Defining Targets

The <target> tag is the real workhorse of an Ant build file. Define a target for each major area of functionality that you need; this generally consists of some common setup, a compilation setup, a step to create a JAR file, and perhaps steps to create a WAR or EAR file.

In our example, we have five targets defined: init, prepare, compile, jar, and clean. The init target in our example merely loads up the properties file, as explained in the previous section. There's nothing stopping you from performing other tasks in this target; one common task that you will find in init is a call to <timestamp>, which sets properties containing the current time and date. You'll see an init target in most build files that you examine.

The prepare target is another common target that provides functionality that can be shared by multiple targets. Here the prepare target is simply creating a directory using the built-in task <mkdir> that we can compile our Java source code in. Notice that the directory we ask it to create is the result of using a property. The <prepare> target could do quite a bit more, such as create entire directory hierarchies, but for our purposes, simply creating this one directory is sufficient.

Notice that at the end of line 8 there's an attribute that says depends="init". This attribute sets up a dependency between the current target and the target(s) specified. When Ant gets ready to run our prepare target, it will see that we have a dependency and will then go off to run the depended-upon

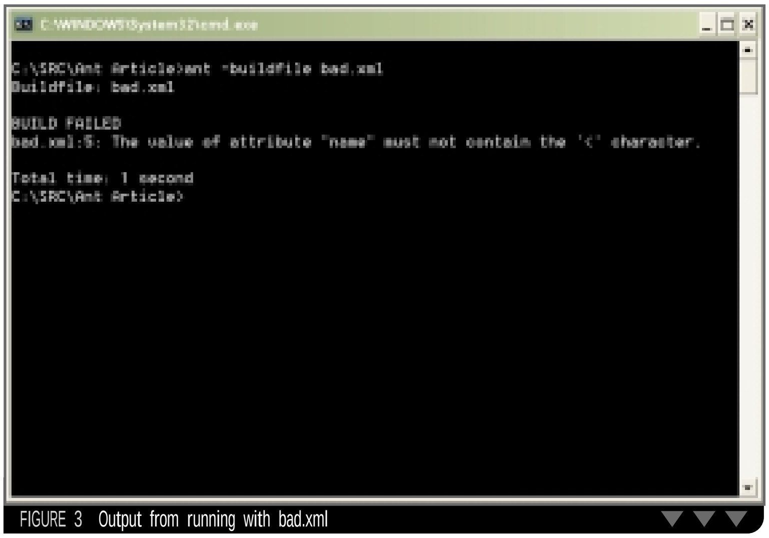


```
C:\SRC\Ant Article>ant clean jar
Buildfile: build.xml

init:

clean:
   [delete] Deleting directory C:\SRC\Ant Article\classes
   [delete] Deleting: C:\SRC\Ant Article\example1.jar

init:

prepare:
    [mkdir] Created dir: C:\SRC\Ant Article\classes

compile:
    [javac] Compiling 1 source file to C:\SRC\Ant Article\classes

jar:
      [jar] Building jar: C:\SRC\Ant Article\example1.jar

BUILD SUCCESSFUL
Total time: 2 seconds
C:\SRC\Ant Article>
```

FIGURE 2 Output from running Ant specifying targets to execute

oracle

www.oracle.com



```
C:\SRC\Ant Article>ant -buildfile bad.xml
Buildfile: bad.xml

BUILD FAILED
bad.xml:5: The value of attribute "name" must not contain the '<' character.

Total time: 1 second
C:\SRC\Ant Article>
```

FIGURE 3 Output from running with bad.xml

target(s) and then come back and run ours. This is a powerful mechanism for avoiding redundancy. As build files get larger, you'll end up with many targets that need some common functionality; moving this commonality into a single target and then setting up a dependency allows this to happen. Here we specify that prepare depends on init, which makes sense because prepare uses a property that's set inside init. If init were not called, we'd end up with a directory called ${build.name} being created. (Yes, we could have put the <mkdir> task directly into the compile target or inside init, but then you wouldn't have seen the dependency mechanism in action.)

Next, our compile target does what you might expect: it compiles our code! As you can see from Listing 1, there's a built-in task called <javac>, a wrapper for the standard javac compiler. It has many attributes, most of which are optional. Here we tell it where to find the source code to compile, srcdir="${src.dir}", and where to put the compiled class files, destdir="${build.dir}". Using the task this way will compile all Java source files that it finds in the ${src.dir} directory (and all of its subdirectories) and put the binary class files in the directory specified by the ${build.dir} attribute. If the source files declared a package, then the package structure will be represented in the destination directory.

Notice that the compile target depends on prepare. Since we need the build directory to be created before we try to write class files to it, we need to call prepare. But we also need the properties that get created inside init. Because compile depends on prepare and prepare depends on init, everything will be executed in the proper order and compile will work just fine.

Moving on, our jar target will take the output of our compile step and build a Java archive out of it. This example is almost overkill because of the simplicity of the project, but I wanted to show you a few of the interesting built-in tasks that Ant provides. You can see here that the jar task will build the archive specified by destfile and will include in it any class files (and directory structure) that are returned by the enclosed <fileset>. In this case, it will return only a single class file, but if there were others located anywhere under ${build.dir}, it would also return those. And notice again that it has dependencies. It wouldn't be terribly useful to create a JAR file without first compiling our source files. Thus setting depends="compile" will cause everything to be compiled (and prepared and inited) before trying to create a JAR. Usually if a target fails, the build aborts. There are a few exceptions, but generally this is the rule.

Finally we have a target called clean. This target can be executed to remove all artifacts of previous builds to ensure a clean build. Here we're deleting the compilation directory and the JAR file created from a previous run. Both these deletes will fail silently if the specified file/directory doesn't exist, which is fine. You'll almost certainly want a clean target in most of your build files, but care must be exercised when using the delete task, the same care you exercise any time you're deleting files and directories. It's quite painful if you delete the current directory and its contents instead of a subdirectory. Notice that our clean target depends on init. The reason for this is that if init is not run, then the two properties that are used by the <delete>s would never be defined and nothing would get deleted.

That's About It

That wraps up our discussion of the simple build file (the Hello, World of the Ant world, if you will). You should now have an understanding of what a build file looks like, how the different pieces relate to each other, how to declare that a task is dependent on another, and what a couple of the built-in tasks are.

When Things Go Awry…

Inevitably you're going to write a build file that you think is perfect, but when you try to run it your screen will fill up with stack traces and other error messages. One of the most common errors in a build file is not properly formatting the XML code. Remember that XML, unlike HTML, requires all attributes to be quoted and all tags must be closed! The build file in Listing 2 will not parse correctly because there are missing quotes all over the place and at least one target isn't closed. Even leaving off one quote will cause your file not to parse properly. An editor that does color syntax highlighting or checks for well-formed XML will help you spot these errors.

Figure 3 shows the output from running Ant with the bad.xml file in Listing 2.

The parser encounters the first error and indicates it as being on line 5 (the number after the colon next to the file name). Actually it's on line 4, which is where we forgot to add the closing quote to the name attribute of the init target. Since Ant couldn't complete parsing the file, it will abort the run

"Inevitably you're going to write a build file that you think is **perfect**, but when you try to run it your screen will fill up with **stack traces** and other error messages"

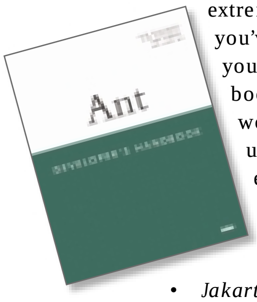
spiritsoft

www.spiritsoft.com

after the first error. Once you add the missing quote and run again, you'll see the next error, an unclosed <property> tag.

Remember that the error messages may not always explicitly help you and, in fact, may be caused by something other than your build file, such as a failed execution of an external program. Get comfortable with Ant's error messages; this is certainly not the last time you'll see one.

Summary



In this article I provided a quick introduction to an extremely useful tool. You should be able to take what you've learned here and immediately start applying it to your current business problem. There are a number of books on Ant available (one of which I coauthored) that would be helpful. Don't forget that the entire Ant manual is included with the Ant distribution and is located in ANT_HOME/docs/manual. It's also available on the Ant home page.

Links

- *Jakarta Web site:* http://jakarta.apache.org
- *Ant pages:* http://jakarta.apache.org/ant
- *Ant Manual:* http://jakarta.apache.org/ant/manual
- *Eclipse:* www.eclipse.org
- *jEdit:* www.jedit.org
- *IntelliJ IDEA:* www.intellij.com/idea
- *BravePoint:* www.bravepoint.com

AUTHOR BIO

Joey Gibson is a senior consultant and instructor for BravePoint, a consulting company in Atlanta, GA. He is the coauthor of Ant Developers Handbook *published by SAMS.*

joey@joeygibson.com

Listing 1: build.xml

```
<?xml version="1.0"?>

<project name="Example1" default="jar" basedir=".">
  <target name="init">
    <property file="build.properties"/>
  </target>

  <target name="prepare" depends="init">
    <mkdir dir="${build.dir}"/>
  </target>

  <target name="compile" depends="prepare">
     <javac srcdir="${src.dir}"
            destdir="${build.dir}"/>
  </target>

  <target name="jar" depends="compile">
    <jar destfile="${jar.name}">
      <fileset dir="${build.dir}"
               includes="**/*.class"/>
    </jar>
  </target>

  <target name="clean" depends="init">
    <delete dir="${build.dir}"/>
    <delete file="${jar.name}"/>
  </target>
</project>
```

Listing 2: bad.xml, A Build File with Problems

```
<?xml version="1.0"?>

<project name="Example1" default="jar" basedir="." >
  <target name="init">
    <property file="build.properties">
  </target>

  <target name="prepare" depends="init">
    <mkdir dir="${build.dir}"/>
  </target>

  <target name="compile" depends=prepare>
    <javac srcdir="${src.dir}"
           destdir="${build.dir}>
</project>
```

Download the Code! www.JavaDevelopersJournal.com

Interview with Microsoft

–continued from page 16

applications reaching over 140 different cell phones and other mobile devices. We work with carriers to enable them to deliver value-added services that developers can leverage, such as location services with technologies like MapPoint .NET. We don't believe there are features in J2ME that are missing in .NET Compact Framework. Quite the opposite, we have a very long list of features in .NET Compact Framework, such as XML Web services support, XML data support, and relational data support, and window UI support that are not supported in J2ME CLDC+MIDP (what people commonly refer to as J2ME), and many of these are not even supported in the J2ME CDC spec (the rarer but larger 2–3MB profile).

<dennis marcum>: **Why can't you just be honest, cut out the marketing nonsense, and let the technologies speak for themselves?**
<microsoft>: As we said earlier, we are happy to talk about the technology. We believe our technology offerings are superior to those of our competitors, and developers who have tried Visual Studio .NET and the .NET Framework seem to agree.

<robert chartier>: **How does Microsoft feel about the U.S. government forcing them to push a competitor's product into the market using their own distribution avenue just because that competitor has chosen to hide behind the law instead of doing something more productive?**
<microsoft>: We believe that once the court has reviewed Sun's extraordinary request for a "must carry" injunction that would force Microsoft to distribute Sun's Java Virtual Machine with Windows, the court will make the correct decision.

<hubert chan>: **I noticed that the new MCSD exams don't include Visual C++ as part of the qualifications. Does this mean that Microsoft is shifting its main development languages focus to C# and VB.NET only? How about the native C++ compiler development?**
<microsoft>: We are committed to the C++ language and are making numerous enhancements to our Visual C++ product. We believe this is an oversight and are moving to correct it.

<malcolm davis>: **Is Microsoft's base strategy of "embrace, extend, and extinguish" still alive or will they follow W3C standards in regards to Web services–based technology such as SOAP? In the past, Microsoft starts off heavily involved and accepting of standards, then works the standards to their own ends.**
<microsoft>: Microsoft has been at the forefront of developing technologies, such as SOAP and WSDL, and then makes them available to the entire developer community. We're committed to supporting the standards for XML Web services and working with the community to develop new ones. That process of extending standards to add new and improved functionality is common in the software industry and a good thing for both developers and consumers. It's fundamental to the design goals of the .NET Framework that interoperability in a platform- and vendor-neutral way is seamlessly possible and we intend to continue to deliver on that.

• • •

For future Ask JDJ sessions and to have your questions answered please check www.n-ary.com/java/jdj/askjdj.cfm.

sitraka

www.sitraka.com

Good News for the Java Universe

Java isn't going away

WRITTEN BY
STEVEN BERKOWITZ

You may have to dig beneath the hype a little, but at any gathering of 40 Java vendors there's bound to be some treasure buried in there somewhere. It's just waiting for you to find it.

If you have children, you tend to measure time by their needs. The first day of summer camp I was at the Web Services Edge 2002 East Conference (hope you found my "Show Report" helpful [*JDJ*, Vol. 7, issue 8]), and the first day back to school found me at Wall Street IT - The Next Generation.

Held September 4 and 5, 2002, at the Metropolitan Pavilion in New York City, Wall Street IT - TNG gathered almost 40 vendors on the show floor and backed them up with 15 seminars and presentations by some high-level players from technology companies and major Wall Street firms. Produced by Lighthouse Partners with show management by Flagg Management Inc., this conference provided a good look at the state of technology.

Don't worry if you're not on Wall Street. The show had plenty of good information and, more important, good news for the Java world. Sure the focus was on using technology in the financial services industry, but many of the vendors work in other spaces as well and these conferences presented a great opportunity to read between the lines. While everyone was waving the Web services banner, by listening closely, I realized something rather valuable: despite how dank and miserable it may be out there, Java is not going away, maybe even can't go away.

Many of the vendors on the floor were Java shops. In and of itself this is neither surprising nor very inspiring. People still need technology, so it's inevitable that some Java work will get done. It's when you look closer that things become interesting.

California-based DevelopMentor provides technical training in Java, XML, Web services, and .NET. They were telling me that at an XML show the prior week, around 70–80% of the show attendees they spoke with were asking about courses on how to do Web services with Java.

Ah. Now we're getting somewhere. If you think about it, trainers have a unique view of the market, especially in tight times like these. Budgets allow for only the most necessary of items. Training courses have to be directly relevant to existing projects.

With this thought in mind, I spent about 30 minutes with Greg Brill, president and founder of Infusion Development Corp. and the editor and driving force behind the CodeNotes book series. Infusion develops custom training courses for corporate clients. Apparently, they're no longer called on for basic Java courses. It's all Java and ___. Clients want a blend of technologies. Infusion recently did a 14-day training course for Lehman in Tokyo – a Java, EJB with WebLogic, and XML/XSL course. Demand for Infusion's courses has remained steady and perhaps even grown a little. This is because their courses are custom-developed to suit the needs of particular projects and, as mentioned earlier, those projects still exist.

What Brill is seeing is that Java is the facilitator of integration; this shows Java's maturity and penetration. In the boom days, Java's hype machine was working full-time: everyone was putting out J2EE servers and everyone else was using them. Now Java is everywhere. It's a fact of life. No matter what happens in the market, no matter what happens with .NET or anything else, Java is not going away. As Brill put it, Java colonized tech companies much as the Spaniards colonized the New World. Spanish is spoken everywhere south of the U.S.

Web services is pretty much guaranteed to play a big part in any IT conference these days. Wall Street IT - TNG did not disappoint. Day one provided us with a seminar given by Anne Thomas-Manes of Systinet entitled "Web Services for Technology Managers," in which she gave a fairly high-level view of the technologies behind Web services and laid out how Web services work. She also discussed which companies are providing Web services solutions and gave some guidance on how to pick from among this field.

As you know, aside from SOAP, WSDL, and UDDI, most of the Web services standards are still evolving. The industry is getting closer to having standards that deal with security, workflow, and transactions, but we're not there yet. This would be a good place to note that the only thing that doesn't seem to move on Internet time is the development of the standards underlying the whole mess.

There seems to be a burgeoning awareness that the exposure of func-

altova

www.altova.com

tionality via Web services, what is basically cross-language RPC, is not all that sexy. According to Thomas-Manes, the best a SOAP message can do is two to three times slower than RMI, and that's top speed on a small, simple message. You'll really need something more to make this take hold after the hype dies down.

There was a panel discussion called "Future Technology Platforms for Deploying Web Services." The moderator was Frank Greco of Crossroads Technologies and the panelists were Jim Bole of Infravio, Dmiitri Tchikatilov of Microsoft, Ed Schwarz of Sun, Adam Greissman of UDICo, and JP Morgenthal of Software AG. It was during this discussion that the "something more" came out. The panel reiterated that Web services are too slow and not secure and that simply exposing old functionality was inadequate.

According to this panel, the real power of Web services is in enabling new semantic definitions of data held in disparate systems. The ability to look at your data in combination, to bring it together, allows you to leverage it in ways not available before. It allows you to create new functionality centered around data access, data management, and business process monitoring. In a presentation immediately preceding this panel discussion, John Stone and JP Morgenthal declared that it won't be mere exposure of existing functionality but rather a massive demand for semantic standards that will ultimately drive interoperability.

This view of Web services is penetrating the marketplace. Based on their Web services management system, Infravio recently deployed a series of applications at an industrial supplies company. Using Java-based Web services, Infravio exposed existing business functionality but then wrote caching code and combined the two to provide real-time access to the company's back-end SAP system.

In talking to Jim Bole, Infravio's VP of engineering, I learned that for his client, the true value of this project was not the exposure of existing functionality but rather the ability to manage the hundreds of services being exposed in this way. While initially this will be an internal facing system only, the company intends to publish it to its suppliers as well. Other values these Web services provide include short time-to-market and a new semantic architecture that allows them to decouple back-end lock in. Let's not forget that it was Java that solved the performance problems by enabling caching. This was a pure Java addition to Web services.

It is this sort of usage that will expand the adoption of Web services and that adoption will spur the standards.

Back in October 2000, Forrester Research coined a new and rather silly term: *X Internet*. There were two presentations that fell under the title "Financial Application Deployment with Web Services and the X Internet." It was during the first of these that Michael Baresich, CEO of CoKinetic, suggested mostly tongue-in-cheek that Forrester came up with this term to sell more reports. Then he defined it: the X Internet is a sloppy term for a thin-client technology that adds more functionality on the client side than is possible with just HTML.

A much better term for this concept is *rich client*. I have Michael Curry, Altio's director of product and services, to thank for that phrase. Such applications bridge the gap between 100% thin-client/HTML solutions and more traditional client/server applications. In your basic browser-based application, what you present to your user is HTML. Doesn't matter how fancy you are on the server side – servlets, JSPs, and STRUTs – the result is a static HTML document. At

jboss

www.jboss.com

precise

www.precise.com

best, this is a compromise. Rich-client applications offer a more robust interface. Curry gave the second X Internet presentation and demonstrated that such applications can offer improved visualization, real-time data (subsecond data updates as opposed to hitting the "refresh" button), and client-side data manipulation.

There are a number of vendors in this product space and they take a variety of approaches. CoKinetic has its CoKinetic Player, which is appropriate for institutional uses. Written in C++, the player is a 2MB download and has a 5MB footprint. However, once installed, ideally as part of the corporate desktop image, it's launched transparently when a user clicks on a link for a CoKinetic application. Synchrony Systems offers Sizzle, a Swing-based rich-client system. Altio wrote its own Java client for the client side of its Altio Live suite of products. All three offer a developer's toolkit for building the user interfaces, and each of these products accepts XML messages to define and update the UI as well as to carry the data to the application.

Altio presents a true Java success story. Its entire product is written in 100% Java. There is the AltioLive Presentation Server that receives messages from your server-side applications and then sends the output to the browser where the AltioLive Smart Client sits. The Smart Client has a 200Kb footprint and handles all the rendering in the browser. Altio chose this approach rather than a more traditional applet for a variety of reasons. First, size. That 200Kb application contains all the UI controls, all the rendering, a DOM and JAX Pack interpreter, as well as the logic for communicating with the server. By building their own client, Altio is able to have better browser compatibility as well. Using the 1.1 JVM, AltioLive Smart Client can work with 4.0 browsers and forward. This way, no one has to download a plug-in to use an Altio application.

I challenged Curry during his presentation, asking how Altio is guarding against backsliding toward the old fat-client problems. His response was that Altio is dedicated to a very strict Model-View-Controller paradigm and the only functionality they put on the client side is the View. Granted, that View is richer than you can get with just HTML, but it is strictly a View. In addition, after the initial client load, the only data that comes down the pipe is XML, so it's easily compressed, and the presentation server needs to send only the data.

Now, being a Java shop, even a good Java shop, is not what makes Altio a success. It's their client base. They've recently installed an AltioLive-based system at The Hartford, a large insurance company. The Hartford has replaced the front end of their old claims-handling system with a rich UI-based system in Altio. The Hartford's primary reason is that they wanted a thin, Web-based client but they needed more functionality than was available in HTML. And the development time with Altio's development environment was much quicker than they would have been able to achieve with a more traditional JSP-based application. Altio is installing systems for seven or eight additional clients who all have similar reasons for choosing Altio – better functionality and faster development.

• • •

Conferences present you with a great opportunity to discover what is happening in the industry. This is why **SYS-CON** was pleased to see how many developers, IT managers, and vendors had attended their Web Services Edge 2002 West Conference & Expo, October 1–3, in San Jose, CA, where – as in New York City – Java-based Web services were very prominent.

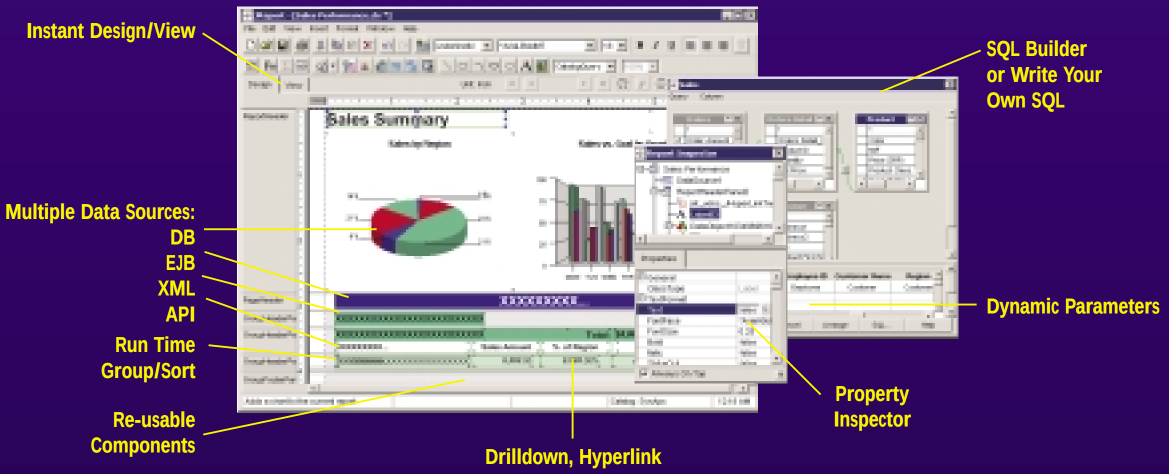
AUTHOR BIO
Steven Berkowitz has done development and project management for Fortune 100 companies, startups, and nonprofit organizations. He recently started techniCrafters to provide Web development services to small business and municipalities.

sjb47@yahoo.com

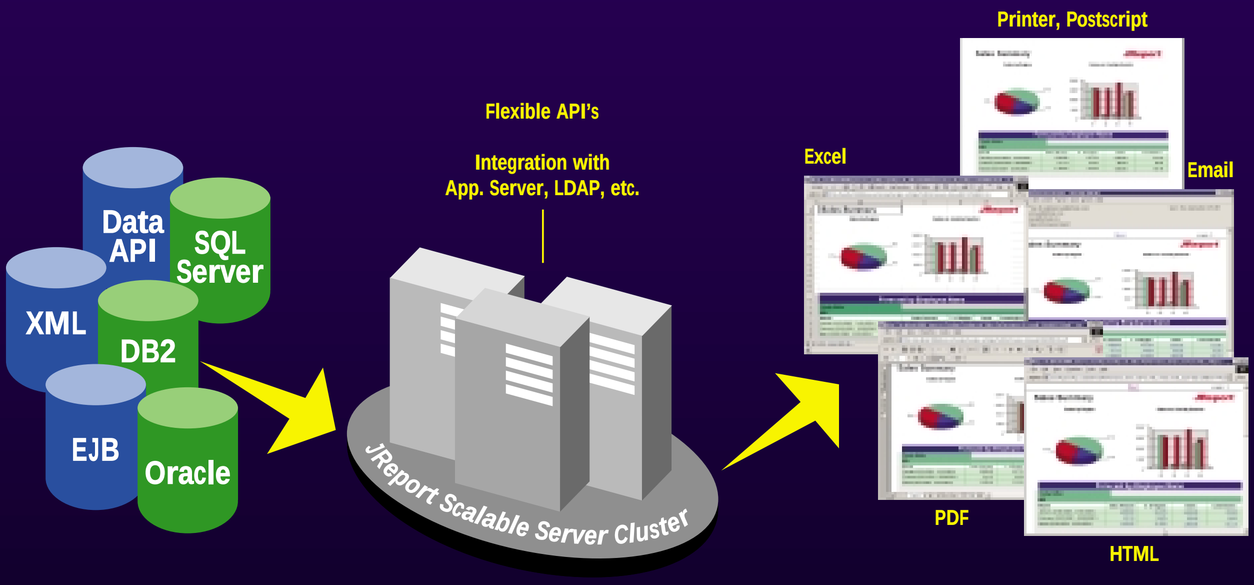
acceltree

www.acceltree.com

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/j11.htm OR CALL 301-838-5560

JASON BELL J2SE EDITOR

What Happened to the Evangelists?

In my last few editorials I've been looking back in order to look forward; for example, how to encourage and empower new programmers, how to learn, and how to create better requirements and user expectations. Now I feel it's time to look forward.

Diving into the dictionary (as I often do), the term *evangelist* is defined as a "bringer of the glad tidings" (*Webster's Revised Unabridged Dictionary*). When was the last time you heard a Java-related story that was going to save your life? Something so radical and amazing that you just had to stop everything and listen. Evangelists have the ability to bring you to the edge of your seat and make you say, "This is for me!" Do you remember 1995–1996, when you couldn't open a computer-related magazine without Kim Polese telling you the benefits of Java and how it would revolutionize the world in a short period of time? Kim also told us about jazz dancing as a way to unwind, but by that time I was off and running and programming in Java. It was probably just as well Kim didn't tell us the news the other way around.

One prime example of this is Java Data Objects (JDO), which stands a good chance of changing the way we access our external data regardless of the data store. Since most people access a relational database with JDBC, not JDO, you need a good evangelist to fight the good fight and tell us why we should change direction. There are times when you need to be motivated to make a radical change and the evangelist knows it – he or she has the passion, the drive, and the tenacity to keep telling about this great new thing until we take notice.

What interested me about JDO was that I have data in databases and in XML that I would like to access the same way as an object. I started reading *Java Data Objects* by Robin M. Roos (Addison-Wesley) and that got me on the right track. The book really motivated me – Mr. Roos described what could be achieved with JDO and I started thinking I could move mountains with this! The only thing that dampened my enthusiasm was the lack of detailed working examples that explained how I could migrate from JDBC to JDO without wanting to give up and look for something else. There was a lot of emphasis on the JDO API, which I don't have a problem with. Perhaps I just need to be motivated differently.

I believe there are so many software vendors offering the same type of product in the name of competition that we shy away from them and stick to what works for us, and business also dictates that there's not much time for research and development, which is also a shame. It makes life difficult for the evangelist and it makes it harder for us to adapt to the evolution of Java.

Christian evangelists found it easy to talk about what they believed in because they believed in it so much; it was so infectious that you had to start questioning whether there was truth in what was being said. The same goes for new technology: Is product XYZ really going to change how I program and improve my day-to-day operations? You have to learn how to weigh what you have heard or read and discern if it's right for you. On the opposite side of the coin, you can tell when someone is trying so hard to sell you a technology they have no faith in – it just never hits the mark.

To survive, Java and third-party APIs and applications need an evangelist to spark our enthusiasm to the same level that got us programming in Java in the first place. If this doesn't happen, we will get deflated over time and move on to something else that fires us up again. Now is the time for new Java evangelists to step forward. Preach it to me brother!

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.

ibm

www.ibm.com

Thread Pooling in Java Applications

What are the risks involved?

WRITTEN BY
Vishal Goenka

There are several textbooks and Internet articles that dwell on the performance and scalability benefits of using a thread pool versus creating new threads in a multithreaded Java application.

While some of them overstate the benefits, most fail to emphasize some of the caveats of Java thread pooling. Due to space contraints, this article provides only a brief summary of the benefits and emphasizes the drawbacks. A list of references that covers the benefits in more detail is provided at the end.

What Is Thread Pooling?

Thread pooling refers to a technique where a pool of worker threads is created and managed by the application. When a new job arrives, instead of creating a new thread to service it, it's queued by the thread-pool manager and dispatched later to one of the available worker threads. The thread-pool manager manages the number of active worker threads based on available resources as well as load considerations, adding new threads to the pool or freeing some worker threads in response to the number of outstanding requests. The primary goals of thread pooling are managing the number of active threads in the system and reducing the overhead of creating new threads by reusing threads from a pool.

Why Pool Threads?

The primary argument in favor of managing the number of active threads in the system is: threads have a memory overhead since each one needs a certain amount of memory for its stack. Threads also add scheduling overhead, since the scheduler's work increases as the number of threads increases. Depending on the implementation of the Java Virtual Machine, each Java thread on certain operating systems may correspond to an OS thread, making Java threads extremely heavyweight, and may limit the total number of active threads that the JVM is allowed to create.

To be clear, I'm not saying you don't need to manage the number of active threads in a system. After all, the benefits of multithreading do have diminishing returns once the number of threads contending for the available CPUs increases. If a server can process only about 1,000 simultaneous requests, it doesn't help to dispatch each incoming request as it's made. Often the requests must be queued and processed at a controlled rate to maintain the number of active requests below the server threshold. A common mistake, however, is to assume that dispatching queued requests automatically calls for the reuse of threads from a thread pool. Dispatching a request to a new thread and letting the thread die once the request is serviced achieves the same effect on managing the number of active threads in the system.

Thread creation also has an overhead that can be higher in many cases than the overhead of managing a thread pool. While the argument still applies, the relative performance impact has changed significantly over the years. The newer JVM implementations are optimized for creating threads; most use a combination of user-level threads (known as green threads) as well as system-level threads (or OS threads) to make creating threads much less expensive than in earlier implementations.

The Dichotomy of Pooling Threads

The reasons for pooling threads seem to make perfect sense, just as connection pooling makes perfect sense in a server-side application. Used inappropriately, thread pooling in Java can introduce serious programming flaws, ranging from logic errors to potential deadlocks and even performance bottlenecks.

I distinguish thread pooling in general from thread pooling in Java simply because many of the arguments that apply to thread pooling in Java do not apply to other programming environments. Perhaps a common source of misconception about the benefits of thread pooling in Java stems from our experiences in other environments where the cost-benefit equation tilts strongly in favor of thread pooling. In the following discussion, "thread pooling" implies "thread pooling in Java," unless stated otherwise.

Thread Pooling Breaks Usage of Thread-Local Variables

Thread pooling is not friendly to the java.lang.ThreadLocal and java.lang.InheritableThreadLocal classes that were introduced in JDK 1.2 to provide thread-local variables. These variables differ from other variables in that each thread has its own independently initialized copy of the variable. The typical usage of a thread-local variable in a multithreaded application is to keep track of some application context associated with the request, such as the identity of the user making the request. The get() and set() methods in the ThreadLocal class return and set the value that corresponds to the executing

crystal decisions

crystaldecisions.com

"Addressing all the design issues that a robust thread-pool library must implement is a nontrivial task"

thread. Thus, each thread executing a get() on a given ThreadLocal variable can potentially get a different object. The set() similarly allows each executing thread to set a different value for the same ThreadLocal variable.

Think of a ThreadLocal variable as a hashmap that stores one value per thread by using the thread as a key into the hashmap; however, these values are "associated" with the thread in a stronger and more intrusive way. Each thread maintains a reference to a private version of a hashmap (implemented as a package accessible class, ThreadLocalMap) that contains all the thread-local variables associated with that thread. Each thread uses the declared ThreadLocal variable as the key into the hashmap to store one value per ThreadLocal variable. When a thread dies and is garbage collected, all thread-local values referenced by it are subject to garbage collection (unless they're referenced elsewhere).

InheritableThreadLocal extends ThreadLocal to allow thread-local variables associated with a parent thread to be inherited by any new child thread created by the parent thread. This class is designed to replace the ThreadLocal in those cases where a per-thread attribute being maintained by the variable, such as UserId, TransactionId, etc., must be automatically transmitted to any child threads that are created. To achieve the inheritance, the Thread class maintains a separate private hashmap (ThreadLocalMap) for inheritable thread-local variables. The Thread constructor ensures that the inheritable thread-local variables of the executing thread (the parent thread) are copied onto itself (the child thread).

Thus, each Thread object has explicit references to all the thread-local variables, which in turn are only accessible via the ThreadLocal or InheritableThreadLocal object. Like normal variables, private ThreadLocal or InheritableThreadLocal variables are only accessible to the declaring class and the threads associated with them. While it's possible to expose a method in the Thread class to "purge" all (inheritable) thread-local variables associated with the thread, it would require additional security checks to ensure that only privileged code can do so, the privilege being ascertained using the Java permission mechanism. Given the lack of such a construct even in the latest versions of the J2SE/J2EE APIs, there's no way for a thread-pool manager to purge or reset all the thread-local variables associated with a given thread when reusing the thread in a different request context without the explicit cooperation of all code that uses any thread-local variables.

Unless the declaring code "removes" a value assignment by explicitly setting the value to null, thread-local variables remain assigned and hence "associated" with the thread. As a result, any code that uses thread locals risks using stale/incorrect values of the variables that were created in an earlier request context when running in a pooled thread. Given that ThreadLocal and InheritableThreadLocal are standard J2SE/J2EE classes, they're quite likely being used in various pieces of library code, none of which is safe to be executed by a pooled thread without an explicit understanding of the usage details.

The only way to get around this is to avoid using a pooled thread to execute code you don't know and control its implementation details. An application that uses a thread pool to dispatch requests made in different contexts is likely to have "inconsistent" logical errors when executing a piece of code while servicing a request that uses a thread-local variable.

Lack of a Standard Thread-Pooling Library

There are several reference/example implementations of a thread pool manager in various texts that describe and prescribe them, but most developers will choose to implement their own since these reference implementations are meant only for illustration and therefore are not product quality, are copyrighted, nonstandard, and often won't meet your specific requirements. Implementing a robust thread-pool library is a complex task that requires extensive tests in a variety of situations, including different operating systems, multiprocessor machines, extensive load testing, various application usage scenarios, and thread-pool management policies. While it seems simple on the surface, a robust implementation must address such issues as pool-size determination based on execution environment and application usage, request throttle, job scheduling, and perhaps even priority scheduling.

When using a new thread per request, the JVM's scheduler ensures that every runnable thread gets a fair share of the CPU, even if the share happens to be really small, as in the case where there are simply too many threads for the given execution environment. Using a size-bounded thread pool can cause queued requests to be starved. If one of the queued requests happens to be a producer (in a typical producer-consumer paradigm), it can lead to a deadlock if all the dispatched requests happen to be consumers waiting for the producer. Such application dependencies may necessitate knowledge of the application logic in the thread-pool dispatching decision, requiring some kind of priority dispatching construct. Priority-based dispatching opens up another can of worms, exemplified by the Mars Pathfinder "reset" problem caused by overlooking the classic priority-inversion problem.

Addressing all the design issues that a robust thread-pool library must implement is a nontrivial task. This happens to be one area of the system that can have systemic effects and bring your application to a grinding halt, unless tested for all potential race conditions and deadlocks, especially since the memory model in multiprocessor systems is often nonintuitive. This is no reflection of your abilities as a programmer, rather a statement about the inherent complexity of the problem and the effort involved in getting a robust implementation.

Performance Benefit Myths of Thread Pooling

While the lack of a standard implementation of a thread-pool library seems like a lame excuse not to use one, it's worth asking why even the latest versions of J2SE and J2EE don't provide one if using a thread pool is so critical to performance on server-side applications. The answer lies in understanding the details of the Java threads implementation. As mentioned earlier, newer JVMs are optimized for thread creation and

macromedia

macromedia.com

destruction and use a combination of user- and system-level threads to minimize the overhead. Not that there aren't any potential benefits in using thread pools, but these are insignificant unless the jobs to be run by pooled threads are short and quick and have a runtime overhead that's comparable to the overhead of thread creation and destruction. Determining the relative overhead of thread creation for the job in question and comparing it with the overhead of thread-pool management must be backed up with real tests in load conditions. As with many performance-related exercises, the results often defy common sense.

To Pool or Not to Pool

Before deciding that you need a thread pool for your application because that little timer thread you need to start for every request seems too much of an overhead, or deciding that you can churn out a thread-pool library for your particular usage in a day or so, here are a few things to consider.

How critical is the performance of that portion of the application and would you make the same decision if it turned out that you needed over a month to write a robust thread-pool library? Is it acceptable to risk an application deadlock due to a less-than-robust thread pool implemented in a few days? Do you have the time to validate and perhaps quantify the savings achieved when using a pooled thread versus creating a new thread? Do you have the time to validate correct behavior under heavy load on a multiprocessor machine, particularly when the boundary conditions on pool size are exercised? If you're not sure about the implementation details of some code, such as usage of thread-local variables, will the pooled thread run it?

In my own experience, a quick and dirty thread-pool implementation of the job at hand often comes back to bite you. A small perceived performance gain is probably not worth the risks introduced by a less-than-robust thread-pool implementation. Not that these concerns don't apply to other design decisions, but thread pooling falls in the category in which the risks are much higher and the benefits are often much lower than perceived.

Summary

Top reasons for pooling threads:

1. Limiting the number of active threads in the system
2. Performance benefits of reducing thread creation overhead

Top reasons for not using thread pools:

1. Breaks usage of java.lang.ThreadLocal and java.lang.InheritedThreadLocal objects
2. Lack of a standard and time-tested thread-pool library
3. The myths of thread pooling performance benefits

References

- Bloch, J. (2001). *Effective Java Programming Language Guide*. Addison-Wesley.
- Pugh, W., Ed. (2001). "The Java Memory Model." University of Maryland. March: www.cs.umd.edu/~pugh/java/memoryModel
- Hyde, P. (1999). *Java Thread Programming*. SAMS.
- Oaks, S., Wong, H., and Loukides, M. (1999). *Java Thread, Second Edition*. O'Reilly.
- Shirazi, J. (2000). *Java Performance Tuning*. O'Reilly.
- Sha L., Rajkumar, R., and Lehoczky, J.P (1990). "Priority Inheritance Protocols: An Approach to Real-Time Synchronization." *IEEE Transactions on Computers*. September.
- Kalinsky, D., and Barr, M. (2002). "Priority Inversion". *Embedded Systems Programming*, April, pp. 55-56.

AUTHOR BIO

Vishal Goenka is a system architect for the core platform components at Campus Pipeline. He holds a BS in computer science from the Indian Institute of Technology, Kanpur (India).

vgoenka@campuspipeline.com

web app cabret

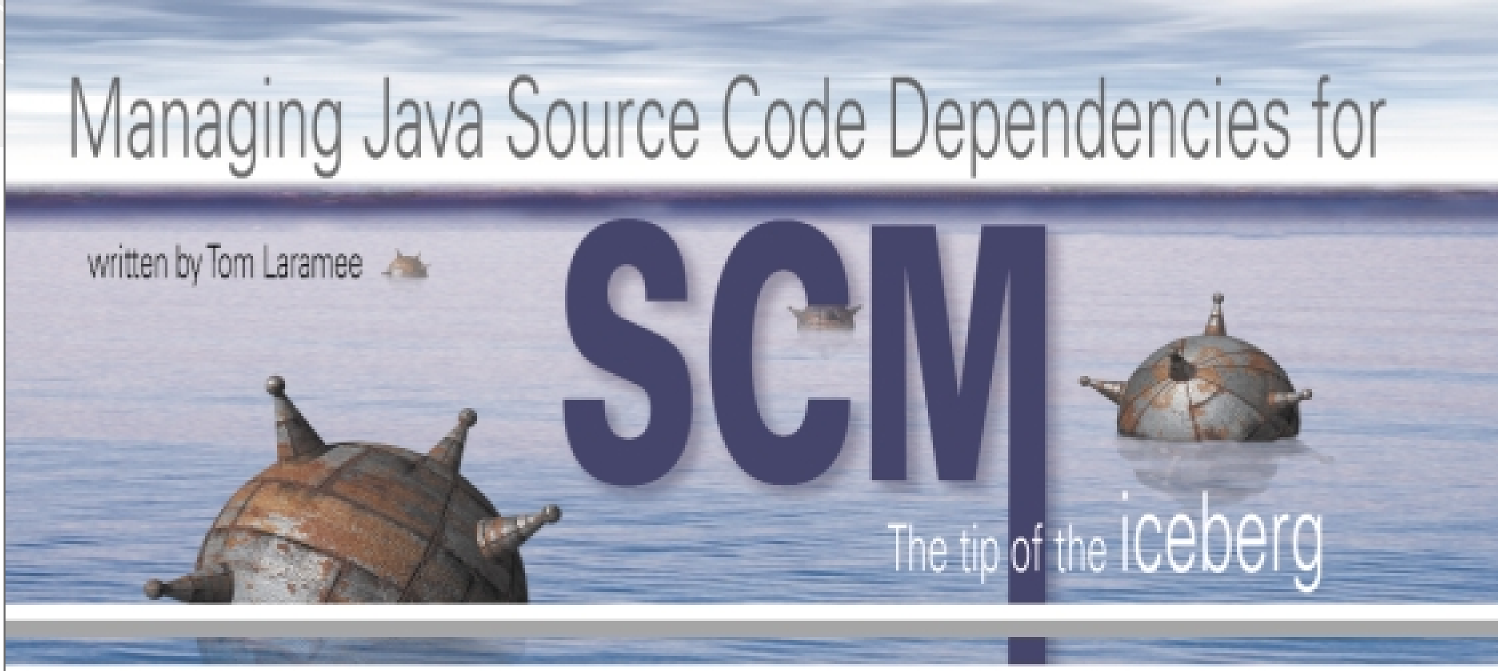
webappcabret.com

parasoft

www.parasoft.com

Managing Java Source Code Dependencies for SCM

written by Tom Laramee



There are many facets to consider when implementing even the most basic software configuration management (SCM). For Java, with its import mechanism, these simple goals often become unmanageable when the source code tree grows beyond a certain point of complexity.

This is mainly due to the reticulate interdependencies that arise within the source code tree as it evolves. Also, because code is seldom (if ever) retired, the code base continues to grow, causing this network to become increasingly complicated over time.

In this article I explore the evolution of the typical Java source code tree and the underlying relationships that make even basic Java SCM problematic. I also suggest a simple way to manage source code relationships to meet basic SCM goals.

Understanding these topics will enable Java development shops to begin implementing simple yet effective SCM systems that balance the requisite process with unencumbered development, testing, and operational deployment. By requisite process I mean staying a couple of steps ahead of SCM-related firefighting while remaining free from laborious and/or unnecessary processes.

Some Simple Goals of Java SCM

- Maintaining source code under revision control
- Managing code dependencies and third-party library dependencies
- Managing builds and build dependencies
- Managing dependencies on third-party JARs

Beyond a certain range of complexity (usually a few hundred total source files, depending on the skill of your developers and how quickly they're being asked to churn out code) the reticulate interdependencies within the code are unable to be unwrapped. That is, the large number of interdependencies introduced by import statements causes artificial dependencies when trying to add features and build, branch, release, and test your code.

More specifically:

- Building a subtree causes the compilation of every source code file in your source tree due to circular dependencies. This results in extremely lengthy build times for some projects I've seen.
- No source code is free to move along under its own development cycle – you might need to build a subbranch *n* times per day and another only *m* times per month, but, because they have import interdependencies, they're both built at the maximum (required) rate.
- Branching and merging are extremely time-consuming and complex and can introduce significant developer downtime, mostly due to the large number of source files that must be considered.
- Releasing code to operations is very difficult, as you have to push every Java class file upon release.
- Testing is more difficult, if not impossible, since it's harder to isolate subbranches of code to understand their functionality. It's also more difficult to write a testing harness for a subbranch (e.g., using JUnit).

Most current source code management tools deal with navigating source hierarchies and finding objects and methods. These are great problems to solve, but not ones that we're primarily interested in (JavaDeps comes relatively close in that it helps to discover some compilation dependencies that go unnoticed by some compilers).

Similarly, many revision control systems (RCS) provide check-in, checkout, branch, and merge capabilities, but none address source code tree structure and how to manage the requisite dependencies involved.

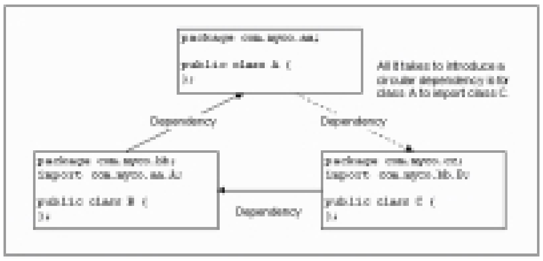
Target Audience

The target audience is developers, testers, and operational support staff who are interested in taking the necessary steps to actively manage their Java-based projects in terms of building source code for test and operational deployment; developing multiple versions of a product or service in parallel; and replicating operational, test, and development environments to reproduce unexpected behavior and fix bugs.

Large numbers of Java source files are in the range of 500..10K with large numbers of dependent third-party JARs in the range of 50..1K. All told, we're talking about a set of development projects that have 0(50L) total document and code artifacts…not very big, but large enough that it's worth exam-

borland

www.borland.com



```
package com.myco.aa;

public class A {
};
```

```
package com.myco.bb;
import com.myco.aa.A;

public class B {
};
```

```
package com.myco.cc;
import com.myco.bb.B;

public class C {
};
```

All it takes to introduce a circular dependency is for class A to import class C.
Dependency
Dependency
Dependency

FIGURE 1: Circular dependencies via Java import statements

ining how the code base evolves and how to keep it from turning into a liability instead of the asset it's intended to be.

Due to its complex nature, this topic is too large to be covered in a single article. I'll start by covering the basics of source code management and builds, and finish by touching on the topics of managing deployments and documentation. Future areas for discussion include managing properties files and build tools and building WAR files.

The Evolution of Java Source Code Hierarchies (aka Back to Basics)

Every Java shop I've ever worked in has followed an eerily similar evolutionary path as far as its Java source code is concerned:

1. Starts the root branch off by creating Java package com.mycompany
2. Begins to populate the source tree with a layer of utility and/or base classes, many of them the usual suspects like com.mycompany.db, com.mycompany.utils, com.mycompany. regexp, and com.mycompany.xml
3. Continues to populate this source tree with a set of servlets, beans, data access, and JSPs that depend on the set of common classes (the aforementioned usual suspects)

This approach is extremely intuitive and works for a while – for about as long as the codebase remains simple enough that dependencies between distinct packages are well understood.

The first dependencies introduced are usually servlets, beans, and JSPs importing common/shared utility classes. These dependencies are distinct, simple, and well understood. However, soon thereafter, more complex references are introduced as developers try to reuse as much code as possible while minimizing the time they spend repackaging code. A servlet from one package begins to look like a utility to another package and is subsequently imported. This type of import can create a circular reference (see Figure 1) between source code files and sets the stage to make even simple SCM prohibitively complex.

Introducing circular references in Java is surprisingly easy and extremely common, though, interestingly enough, I've never actually heard of a developer admitting to such a practice. Understanding these relationships, plus your source code's dependencies on third-party JAR files, is key to having a modular, branchable, buildable, testable, and deployable codebase.

Partitioning Source Code: The Introduction of Components

The first step in decoupling direct source code dependencies is to partition your source code into components.

A component is a set of Java packages that provides a specific set of functionality and has its own development cycle. It doesn't matter whether it's one package or 20, one source file or 200 source files (though using more than a few hundred source files in one component will bring you right back where you started, in terms of problematic source code management). Having their own development cycle means that, relative to the other components, the source files need to be built/tested/deployed *n* times a day while other source code needs to go through this cycle *m* times a day.

Partitioning source code into components will become fairly intuitive after a few examples:

- ***Example 1***

  Utils make great components because they are shared by so many other source files and therefore are dependent on a lot of files. This also causes them to have a quicker dev cycle (and therefore a quicker build turnaround) than most other source code. Create one component for all your utils, or partition them further into multiple components (see Tables 1 and 2).

- ***Example 2***

  Database access classes can be grouped into separate components. For multiple database servers, use multiple data access components, tying each schema to a component 1:1. This handles schema changes nicely and helps manage a component's dependencies on multiple database servers (see Table 3).

- ***Example 3***

  A set of JSPs or servlets that provides a specific set of functionality should be a separate component. This could be a data-entry application, a data-feed reader, or an administrative UI for one of your internal systems. Because these types of components have their own requirements and delivery dates and the requirements change, they end up on their own development schedule, so it makes sense to create a component here (see Table 4).

You could end up with as many as several hundred components, each with anywhere between 10 and perhaps 350 source files. Although partitioning your source code looks complicated, it's actually easy (the difficult part is getting your builds started).

All this source code needs to be checked into a revision control system (RCS). Any/all RCS syntax in this article will be in reference to Perforce (www.perforce.com), as it is has many features that make SCM very simple.

In Perforce parlance, the component source code is checked into location:

```
//depot/components/<component_name>/src
```

"Every Java shop I've ever worked in **has followed** an eerily similar **evolutionary path** as far as its **Java source code** is concerned" nsoftware

www.nsoftware.com

For example:

```
//depot/components/FileUtils/src/com/mycompany/...
//depot/components/DataParser/src
//depot/components/UserData/src
```

Builds, branches, and documentation are also partitioned under each component for RCS:

```
//depot/components/<component_name>/src
//depot/components/<component_name>/branch
//depot/components/<component_name>/builds
//depot/components/<component_name>/docs
```

Third-Party JAR and ZIP Files

The other source for build dependencies are between your source code and JARs provided by a third party.

This necessitates actively managing these files to keep on top of their multiple versions and frequent name collisions. It's very easy to impede the progress of debugging and building through the mismanagement of third-party JARs and ZIPs (e.g., opening up JARs manually to try to find a version number to find out what you built against, or what version you have in production), and yet remarkably simple to organize them intuitively and efficiently.

Because successive versions of third-party JARs sometimes result in name collisions, it's necessary to use the version numbers to maintain them under RCS. In Perforce, the JARs might look like the following (using the JDK and JSDK as examples):

```
//depot/jars/jdk/1.2.2/rt.jar
//depot/jars/jdk/1.3.0/rt.jar
//depot/jars/jdk/1.3.1/rt.jar
//depot/jars/jsdk/2.0/jsdk.jar
//depot/jars/jsdk/2.1/server.jar
//depot/jars/jsdk/2.1/servlet.jar
//depot/jars/jsdk/2.2/servlet.jar
```

This versioning scheme allows components that might depend on the 2.1 version of servlet.jar to reside next to components that might depend on the 2.2 version. Both components can be built and deployed in parallel and their dependencies tracked accordingly.

This approach also has the added bonus of allowing for any client that has access to your RCS server to be able to run builds, as every server has access to the requisite JARs via RCS.

Building Components

Now that your source code is partitioned and third-party JARs are under RCS, it's time to start building. Build requirements are very simple:

- A build for one component may only execute against that component's source code. All other build dependencies must be linked through other components' builds or third-party JAR/ZIP files. In short, a component build may not execute against any source code other than its own.
- Results of builds (JARs) must be under RCS.
- Source code needs to be labeled with the build number, so there is a link between a build JAR and the source code that produced that JAR/WAR. This implies that given any JAR for any component, the original set of source code can be located.
- The dependencies for a deployment (a set of JARS that are deployed together into QA/dev for testing/production) must be under revision control; i.e., the list of dependent JARs for a build of a component must be under revision control.

This first component built must be entirely self-contained – it can be built using only its own source code and (optionally) third-party JAR files. Components built this way are seed builds and start your build process. Build each of these components one at a time by compiling their Java source, JARing up the resultant class files, and checking these JARs into your RCS (build scripts should do all this for you).

If you can't isolate a component so that it's entirely self-contained, either repackage your source code (not often done due to time constraints) or generate an invalid build so you can begin to generate seed builds. (An invalid build is when a component is built against its own source code plus the source code of another component. Sometimes it's impossible to isolate even one component so it's self-contained, so you'll need to build it against multiple components' source code to get started. After this initial build, you'll be able to build it against its own source code and JARs created from this first build.)

A high-level overview of a build involves the following steps:

1. Sync up source code and third-party JARs from your RCS to your local machine.
2. Make sure your target build number hasn't been built already.
3. Set up your CLASSPATH, which contains three sets of entries:
   - The path to the root of the component's source
   - The paths to other JAR files from other components
   - The paths to requisite versions of third-party JAR files

Component	Java Packages
utils	com.mycompany.utils com.mycompany.utils.email com.mycompany.utils.file com.mycompany.utils.network

TABLE 1: One component

Component	Java Packages
utils	com.mycompany.utils
email_utils	com.mycompany.utils.email
xml_utils	com.mycompany.utils.xml

TABLE 2: Multiple components

Component	Java Packages
userdata	com.mycompany.db.users
customerdata	com.mycompany.db.customers
inventorydata	com.mycompany.db.inventory

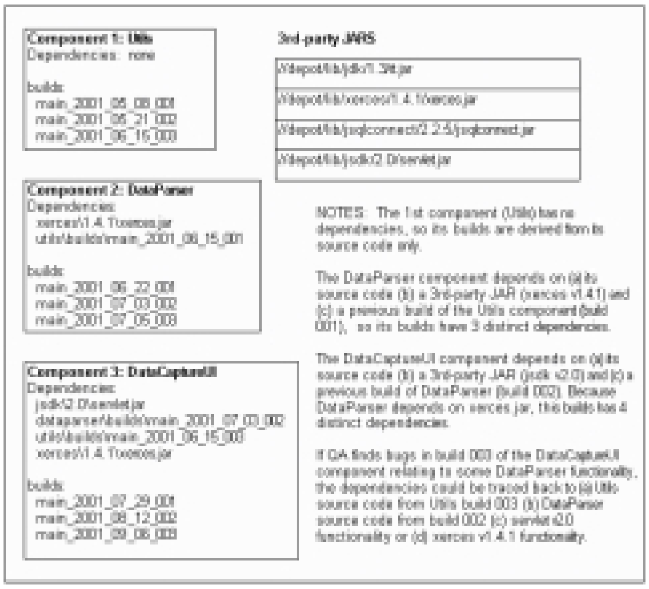
TABLE 3: Multiple data access components

Component	Java Packages
AdminUI	com.mycompany.servlet.admin
DataEntry	com.mycompany.servlet.dataentry com.mycompany.servlet.dataentry.parser
DataService	com.mycompany.servlet.dataservice com.mycompany.servlet.dataservice.cache com.mycompany.servlet.dataservice.interface com.mycompany.dataservice.factory com.mycompany.dataservice.threadpool

TABLE 4: Separate components

engenuity

www.engenuity.com



NOTES: The 1st component (Utils) has no dependencies, so its builds are derived from its source code only.

The DataParser component depends on (a) its source code (b) a 3rd-party JAR (xerces v1.4.1) and (c) a previous build of the Utils component (build 001), so its builds have 3 distinct dependencies.

The DataCaptureUI component depends on (a) its source code (b) a 3rd-party JAR (jsdk v2.0) and (c) a previous build of DataParser (build 002). Because DataParser depends on xerces.jar, this builds has 4 distinct dependencies.

If QA finds bugs in build 003 of the DataCaptureUI component relating to some DataParser functionality, the dependencies could be traced back to (a) Utils source code from Utils build 003 (b) DataParser source code from build 002 (c) servlet v2.0 functionality or (d) xerces v1.4.1 functionality.

FIGURE 2: Build example with dependencies

4. Execute make to build your source.
5. JAR up the resultant class files and check this JAR into RCS.
6. Generate a build summary file (containing the environment, date, etc.) and check this into RCS.
7. Generate a label and stamp the source code for the build with it.

Once you have all of your seed builds, begin to build those components that have only one level of dependence on other source code within your repository, i.e., they can be built using only their own source code, the seed JARs, and, optionally, third-party JARs. Build each of these components individually, JAR up their resultant class files, and check these JARs into RCS.

Once all your one-level dependence builds are complete, it's open season to build the rest of your components, usually done in the order of increasing number of dependencies. The goal is to make sure no component is compiled against any source code except its own.

What you're effectively doing here is isolating like branches of Java code in sets of Java packages against changes in other branches of Java code (also grouped in Java packages). This is probably the most important aspect of the build strategy. This allows stakeholders of your SCM system to isolate, and therefore understand, the dependencies between source code and JARs, and trace any build to the source code that was used to generate that build as well as replicate an environment by easily reproducing the JARs used to construct the environment.

Equally important is that the source code for a component is associated with its build JAR via a label, so it's easy to trace any class file you have in production back to its source files, and then from there to trace other dependent components' class files back to their corresponding source code.

This method of organizing your builds also frees up any components that share a dependency on a common component (e.g., utils). The common component is now on its own development cycle, so it can iterate through many build cycles while allowing dependent components to migrate to newer builds when it makes the most sense. Said another way, it allows for independent/parallel development of components that have a dependency on a single, shared component.

Build Example

At a high level, consider the following scenario for building your first three components:

1. ***Component 1 – Utils:*** No dependencies. Build it against its own Java source code to generate a JAR file that contains the resultant .class files.
2. ***Component 2 – DataParser:*** Depends on a previous build of the Utils component, as well as a third-party JAR called xerces.jar (v1.4.1). Build it against its own source code, a previous Utils component build, and the xerces.jar file from xerces v1.4.1.
3. ***Component 3 – DataCaptureUI:*** Depends on a previous build of DataParser, a previous build of Utils, and a third-party JAR called servlet.jar (v2.0). Build it against its own source code, a previous DataParser component build, a previous Utils component build, and the servlet.jar file from jsdk 2.0.

Note that because Component 3 depends on DataParser, and DataParser depends on xerces.jar, you'll need to add xerces.jar as a dependent JAR for the DataCaptureUI build.

The above set of builds and dependencies is shown in Figure 2.

Conclusion

Managing source code dependencies is only the tip of the iceberg for comprehensive SCM. Other facets of SCM that fit into the component model include:

- ***Managing deployments:*** A deployment is the set JAR, ZIP, WAR, and properties files that allow the component to operate in its designated environment (usually dev, test, or operations). Property and config files can be partitioned similar to source code, whereupon all component artifacts can be synced directly from your RCS server to their deployment server with deployment dependencies tested and well understood.
- ***Managing documentation:*** Component documentation can be bundled with its corresponding component under RCS and mapped to a mount-point on your intranet server for automated publishing. Documentation management has a large number of implicit requirements involving availability, content, and versioning from release to release.

Partitioning Java source code into components and formalizing dependencies will provide several key benefits for your Java-based projects, some of which are implicit thus far:

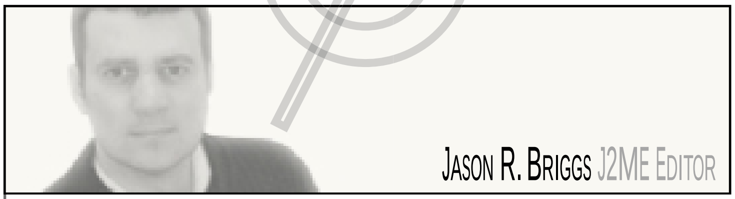
- Ability to provide parallel development of projects that share a common codebase
- Ability to easily deploy to development, test, and operational environments
- Ability to minimize the amount of code associated with a build/deployment
- Elimination of confusion and name collisions due to third-party JAR dependencies
- Reproduction of deployment environments to help reproduce problems (and then eliminate them)
- Ability to retire code and branches of code when a component is retired

AUTHOR BIO
Tom Laramee is a software developer currently working with the Blindsight Corporation writing computer vision software for embedded systems and handheld devices. He has spent the last five years designing and building Web applications as both a development lead and system architect. He holds an MS in electrical and computer engineering from the University of Massachusetts, Amherst.

laramee@pobox.com

new atlanta

newatlanta.com



JASON R. BRIGGS J2ME EDITOR

Has Hell Frozen Over?

It's just as well I'm not a gambler. After pessimistically deciding that it would be a clichéd "cold day in hell" before I saw a Java-enabled phone arrive on these shores, our local Vodafone launched the excellent Nokia 7650. Color and Java, no less. Of course, certain international readers will now be yawning because they've had Java phones for a while, and color for a proverbial age. If you happen to be reading this in Japan, you're probably wondering what the fuss is all about.

Trying to get an evaluation 7650 out of Nokia is about as easy as removing a sore tooth from a wide-awake Siberian tiger just as a veterinarian pokes a cold rectal probe in a sensitive area. Actually, no, scratch that. Considering the glacial lack of reaction from Nokia, perhaps a better description is trying to get served in a high-class restaurant when a movie star has just walked through the door. If you still don't understand what I mean, think about the following statement: "In space, no one can hear you scream…."

It appears that the same is true when you're trying to get the attention of a phone manufacturer.

It seems that our other major telco has decided to look into J2ME's competitor in the mobile space: BREW, with a comment, attributed to this telco's management, along the lines of "…applications can get approved by Qualcomm for a few hundred dollars."

Which is where they lost me. After looking on the Qualcomm/BREW site, the best I could come up with was about $1,150 to certify yourself as a developer and then have a minimal (in their words, Tier 1) application certified.

There are a couple of ways of looking at this cost. From one point of view, a centralized certification process means that a telco can be relatively confident that an application will not cause problems on the phones of their client base. From another point of view, Java's inherent security – and the limitations that have been built into the MIDP platform for that very reason – does remove most of the necessity for an intensive certification process and for digging rather deeply in your wallet to get your newly developed app onto phones.

My belief is that one method will tend to foster an independent and competitive developer community (as well as commercial development companies), and the other will only favor those with money to spend. I also believe that the most innovative applications are going to come out of that independent developer community. This doesn't even take into consideration the likelihood of obtaining open-source applications if developers have to dish out their hard-earned cash to get onto a phone. I'm interested in hearing what other people's opinions are in that respect.

The device market is heating up for Java-capable hardware. If you haven't yet found Java's wireless device page (http://wireless.java.sun.com/device), it's worth taking a look. The list is already huge, and I don't believe it's complete (they're missing the Nokia 6650 to start with). Despite this ever-growing list, there's still a lack of education among the general public as to what J2ME is and why they need it on their phones. This is partly the fault of Sun and the device manufacturers in their efforts (or lack thereof) to publicize the technology in a way that is appealing to the masses.

However, they're not entirely to blame – perhaps the main reason is that there is still no killer app out there that's a "must have." What we need is an app that generates its own "word-of-mouth" buzz – it'll probably be aimed at the teenage market and, undoubtedly, will be entertainment-oriented. This is complete conjecture, but this killer application will be what finally drives any explosion of J2ME phone sales. Perhaps you're developing it right now. Let me know.

• • •

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

This month, Bill Ray investigates encryption on limited devices, and Roger Ritter looks at what you can expect from the second iteration of the MIDP specification. Roll on MIDP 3!!

qualcomm

qualcomm.com

Unlimited Encryption on Limited Devices

Security on your mobile phone

WRITTEN BY
BILL RAY

I have the dubious honor of having written one of the very first implementations of the RSA cryptographic algorithm in Java some years ago, and very badly I wrote it too.

With a 4-bit key it worked great, with an 8-bit key it took about 30 minutes to encrypt or decrypt anything, and after three days of trying with a 16-bit key, we had to use the computer for something else. Just to give you some idea, even back then 128 bits was considered the minimum for secure communications, and each bit doubles the time. Cryptography is not fast; its security is bound up in the complexity of its algorithms. Those who are writing modern cryptography need to be much better mathematicians than I. Of course, Java is not a fast language – as Java developers, the price we pay for platform independence and stability is speed – so writing cryptography in Java doesn't make a lot of sense on the surface.

On servers it's a relatively simple matter to add another processor, and even today's desktop systems have no problems running (efficiently coded) cryptography routines in Java, but mobile devices have enough trouble just updating the screen and responding to inputs. Getting Java working is hard enough in such constrained environments, but adding cryptography to the Java mix on a mobile device is surely madness!

Mobile Security

Getting decent cryptography onto mobile devices has been an aim for a long time; while SSL (or TLS, as it's now known) provides for most of our security needs on the desktop, the algorithms and processes it uses are often beyond the ability of the devices in our pockets. Mobile telephones, in particular, are changing into mobile payment devices in the first stages of the long-awaited move to an electronic currency. PayBox is already offering a service enabling payment, both on- and offline, via a GSM mobile telephone, while car-parking meters in the north of England can be reset from a mobile phone. Users are realizing that their mobile phone can do a lot more for them, but current systems are clumsy and often expensive to run, not to mention that when I want to buy something from a shop, it's insane that the shopkeeper and I both have to make phone calls to a central server to authorize the payment.

Technologies like Bluetooth are providing a conduit for more direct interaction – a mobile wallet in a Bluetooth-equipped mobile telephone could be used to pay for goods in shops, and even automatically authorize payment for transport without user involvement. In this environment, mobile telephones have one great advantage over PDAs: they're always on, ready to respond to incoming requests or scheduled events.

The barriers to such usage, while temporary, are considerable. The cost of an infrastructure to implement such a payment system is massive, but the lack of standardization is the primary problem. Once everyone is using a standard payment system (perhaps a new Bluetooth profile?), we'll see a mass deployment of mobile payment systems; until then they'll be limited to corporate installations and arcologies, but even then only if strong encryption can be deployed on devices like mobile phones and is available to application developers. GSM mobile phones have a Subscriber Identity Module (SIM) that is often capable of dealing with strong cryptography, but it's also under the complete control of the network operator (for sound commercial reasons) and is rarely available to anyone else.

Why Use Java?

Speed is not the only thing that counts against Java when considering cryptography; memory safety is another issue that has never been satisfactorily resolved. Cryptography is generally based around keys, and security is managed by limiting access to certain keys and ensuring those keys are defended against attack. Applications like Pretty Good Privacy (PGP) store the keys, which are encrypted using a passphrase as a key to decode them, on a desktop computer's hard drive; but while the keys are being used they are in memory and could, in theory, be accessed by another application running at the same time, depending on the operating system being used.

In C it's possible to take steps against such theft, but in Java the programmer has almost no control of the memory, and keys held in RAM could conceivably be read by another application. Even worse, if the encryption program is relying on Java garbage collection, there's no telling how long those keys will actually exist in memory; they could even find themselves paged into virtual memory (on the hard drive) where they might hang about for months before that part of the hard drive is used again!

Java implementations of cryptography generally recommend not using virtual memory, and a modern desktop operating system should provide some memory protection to separate processes, but handheld systems rarely have such protection. Even though being paged to a hard drive isn't a problem, having keys hanging about in memory is not an ideal situation.

Working around these problems isn't easy, so there must be a significant advantage to make Java worthwhile, and it is platform independence that provides that advantage. On desktop computers we have to deal with two or three different operating systems. Even on PDAs there are only three or four to worry about, but on mobile telephones things are a lot more complicated. Symbian has taken great strides with EPOC (their OS), but Palm and WinCE are vying for some market share, not to mention the half-a-dozen proprietary operating systems that are still being used. Even code written in ANSI C can't be relied on to work with every mobile phone. Java is rapidly emerging as the sitraka

www.sitraka.com

true cross-hardware application platform. Many phones don't actually have the capability to install and remove applications, but even those are moving toward being able to execute MIDlets, opening Java to everything but the most basic handsets.

Java cryptography certainly would be useful for limited devices, and the Connected Limited Device Configuration would seem ideal, but getting RSA, or something similar, working at an acceptable speed would be close to impossible…which brings us to NTRU.

A Different Solution

NTRU was started in 1996 to capitalize on the development by the founders of a new encryption algorithm designed to minimize processing requirements and run on limited devices. In the field of cryptography there are many companies offering "secret" algorithms that they claim are breakthroughs. However, we generally take "secret" to mean "untested," as peer review is the only way for an encryption system to be proved secure. It is, of course, impossible to prove that an encryption system is secure; you can only prove that you personally can't break it (which, if you are me, is no great recommendation). When looking at an encryption algorithm it's important to know that reputable people in the field have attacked it and can't break it. NTRU has spent a few years doing that, relying on a patent to protect its IPR and working with the cryptography industry to demonstrate the strength of its algorithm.

Even with a much-improved algorithm, getting something small enough and fast enough for CLDC devices isn't easy. Trying to do it in Java isn't likely to make it any easier. Even NTRU didn't start in Java; it offers implementations in a variety of languages and on some very small devices, including RFID tags and Smart Cards. But as already discussed, it's the mobile phone explosion that presents the most interesting and potentially profitable application of encryption – and Java is the only effective way of reaching those platforms.

Having decided to work on a Java implementation of NTRU's algorithm, there's the minor matter of which version of Java to support. The days of just being "Java compatible" are long gone, and now there seems to be almost as many versions of Java as there were programming languages that Java was supposed to replace! Most of the mobile phones support at least the CLDC specification, so that was a logical place to start.

The NTRU algorithm doesn't require floating point mathematics, which helps, and being an encryption library it has no graphical requirements so it should be usable on all versions of Java, once developed for the CLDC. Indeed, the same core encryption and decryption code is used on both the server and client sides (the server is designed to run under J2SE and provides the same cryptographic services as the client).

As the algorithm had already been implemented in several languages, the port of the core code to Java was completed in a few months by a small team of developers, though testing and integration took considerably longer. Embedded developers still have very little choice in terms of working environment, so the Wireless Toolkit from Sun represents state-of-the-art. NTRU uses Visual Café for its server-side development, as this appears to produce faster compiled code, but on the CLDC side, text editors and command-line compilers are the order of the day.

Worse is the lack of debugging or remote testing environments. While desktop objects may contain their own test harnesses or be loaded into test containers, when your whole application is supposed to be under 5KB it's not easy to get testing in there too. In fact, when you're working under such constraints there are good arguments against object-orientation, and the core of NTRU's cryptographic code reflects this philosophy. By explicitly creating a memory context and managing variables within that context, it hopes to avoid the persistence problems inherent in Java applications, and in most cases it should be right. By avoiding object methodologies, NTRU can also avoid dependence on the Java garbage collector, which can be an unreliable beast at the best of times especially on embedded platforms.

NTRU also decided not to support the JCE. The Java Cryptographic Extension is a mechanism that allows suppliers of cryptography to integrate their libraries in a standard way with Java applications. The API is certainly flexible, allowing detailed control of the cryptographic process, but some would claim that flexibility has led to excessive complexity and made the API difficult to use. People who work in cryptography regularly might find the JCE pretty intuitive, but Java programmers just want to be able to encrypt and decrypt without worrying about the messy details. There is the argument that such programming should only ever be done by professionals, but back in the real world we all want a simple API. There's also the issue that the JCE is not supported by the CLDC, and implementing it would have made the NTRU code much bigger.

Getting cryptographic routines into a 5K CLDC–compatible library is very impressive, but it's not going to solve most of the problems with the scenarios discussed at the beginning of this article. The ability to encrypt communications and provide digital signatures (to enable proof-of-identity, essential for any m-commerce system) is only part of the problem; there's still the issue of where and how the keys will be stored. There is no point in providing wonderfully secure communications when the encryption keys have to be downloaded over an insecure link, so keys will have to be stored somewhere on the device and in a secure manner.

Most of the J2ME mobile telephones being launched at the moment have only the ability to run downloaded Java code. MIDP application installation and management is generally kept to a minimum to ensure the simplicity of the user interface, and file management is often nonexistent. MIDP specifies that data can be stored locally, but the developer has no idea where (physically) that data will be stored and so can't rely on it being secure. NTRU is in the process of approaching handset manufacturers about finding some secure space on their handsets for key storage, but again the lack of standardization causes problems. Manufacturers will need a lot of convincing to embed facilities for NTRU in their handsets, especially when NTRU will be collecting a license on every device using its algorithm. Not to mention, once we're talking about putting things into the handsets, we might as well embed some cryptographic hardware.

Doing the encryption in hardware is inherently more secure than leaving it to the software, and this issue is not limited to mobile devices. As the recent releases about Palladium make clear, the only way to secure any system is to embed the cryptography at a hardware level. GSM mobile phones have a cryptographic chip already embedded in the form of a SIM chip, and if we're going to add cryptographic functions, this is the sensible place for them. While the NTRU algorithm may well be a revolution, and getting it working in Java is spectacular, the practicality and usefulness of a layered security model remain to be seen. We'll certainly see the NTRU algorithm in many places, but it's still hard to see Java being one of them.

bill@network23.co.uk

AUTHOR BIO
Bill Ray has worked for several telecommunications companies around Europe, including Swisscom where he was responsible for the development of their Java-compatible DTV platform. He is security editor for Wireless Business & Technology *and coauthor of* Professional Mobile Java Development, *published by Wrox Press.*

inetsoft

www.inetsoft.com

MIDP 2.0 mobile computing arrives

Written by Roger Ritter

Create fully functional mobile data systems

Several years ago Motorola, Inc., and Sun Microsystems, Inc., recognized a potential new market for the Java programming language. Small mobile devices, such as cell phones, were becoming more powerful but did not provide a common programming platform. With different processors, operating systems, and capabilities, it was impossible to write an application that would work on more than one family of devices.

This situation is ideal for Java – its interpretive nature hides hardware differences and provides a single, consistent set of APIs for developers to write to. The only problem was that Java was big, too big for the typical cell phone, whose memory is measured in kilobytes rather than the megabytes that the Java Virtual Machine and associated APIs needed.

To meet this new market, the two companies started a development program to trim the JVM until it could fit into limited, battery-powered mobile devices. Once they proved that it could be done and demonstrated it at JavaOne in 1999, they decided to standardize it through the Java Community Process. This effort produced the Connected, Limited Device Configuration, CLDC 1.0, and the Mobile Information Device Profile, MIDP 1.0, both of which are part of the Java 2 Micro Edition (J2ME).

MIDP 1.0

MIDP 1.0's goals were simple: define a small Java profile that would run on a variety of small devices. At a minimum, the devices would have a screen size of 96x54 pixels, with one-bit display depth and approximately square pixels. They could use a keyboard (either QWERTY or phone-style) or touch screen for user input and have two-way, possibly intermittent, wireless networking. Memory was quite limited, but would include nonvolatile storage for user applications and their data. The Java classes defined by the MIDP 1.0 specification were sufficient to develop small applications, but were quite limited in their display and interaction capabilities.

Many developers recognized these limitations almost immediately on release of the final specification and started asking for changes. Among other limitations, the MIDP 1.0 user interface classes (called, together, the LCDUI) defined only a small selection of user interface components, and this selection wasn't extensible. There was no way to position the elements, so there was no way to group elements on screen. The specification required MIDP-compliant devices to support HTTP networking, but e-commerce applications need HTTPS for security. There was also no security differentiation for MIDP applications (MIDlets). Although the Java sandbox security model provided security from rogue MIDlets, MIDP 1.0 did not provide a way for a device manufacturer or service provider to define trusted MIDlets that could be given greater access to a device's capabilities. Finally, the sound capabilities defined by the first MIDP specification were limited.

All these limitations were required by the restricted capabilities of cell phones, PDAs, pagers, and other small portable devices in development during the original MIDP committee meetings. Since the hardware has been improving rapidly, a follow-up committee was quickly established to expand the MIDP capabilities to match new hardware and to address the concerns and problems identified in the first specification. The new capabilities defined in the next-generation MIDP specification (MIDP 2.0) include new graphics and user interface classes, secure communications, a MIDlet security model, better sound capabilities, and enhanced communications methods.

The MIDP 2.0 committee recognized from the beginning that the new MIDP specification must remain compatible with MIDP 1.0. The first MIDP specification was limited, but it was also popular. By the time the MIDP 2.0 specification would be ready, there would be hundreds of thousands, perhaps millions, of MIDP 1.0 devices on the market. It wouldn't be practical to declare them obsolete and start over. The committee

altoweb

www.altoweb.com

decided that MIDP 2.0 would supplement, not supplant, MIDP 1.0. Device manufacturers and developers could continue to develop for MIDP 1.0 if they didn't need the newer features, or could design to MIDP 2.0 requirements for higher-end devices and applications.

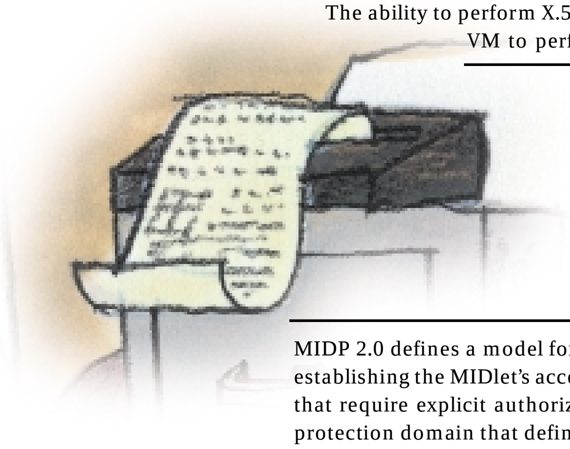
Let's look at the new features.

Security

Probably the most necessary of the higher-end services is secure HTTP connections. Many of the applications proposed for MIDP-class devices are e-commerce applications – simple things like the ability to buy movie tickets online while traveling to the theater. Other applications that transmit personal data or produce billing information also require secure communications. Although these applications are possible without secure HTTP, most people don't want to risk their credit card or personal information over insecure network links. In real life, HTTPS is a requirement for any application that expects to send secure information to a Web site.

MIDP 2.0 satisfies this need with several new components. The HttpsConnection interface defines the necessary methods and constants to establish a secure HTTP connection in accordance with RFC 2818. The connection can be established in several ways. It can use HTTP over TLS, SSL 3.0, WTLS, or WAP TLS Profile and Tunneling. Information is authenticated through the use of X.509 certificates. At a lower level in the network stack, a SecureConnection interface provides the secure socket connection for those applications that want an SSL connection without using HTTPS. In addition, a SecurityInfo interface allows the application to access information about a secure connection. This information includes details of the protocol used, the cipher suite, and the certificate that authenticates the connection.

The ability to perform X.509 authentication also allows the VM to perform authentication on MIDlets.



"In real life, HTTPS is a requirement for any application that expects to send secure information to a Web site"

MIDP 2.0 defines a model for identifying trusted MIDlets and establishing the MIDlet's access to particular APIs or functions that require explicit authorization. This model begins with a protection domain that defines a set of permissions and related interaction modes. The permissions are divided into two sets: allowed and user. Allowed permissions don't require user interaction and provide access to protected functions within the protection domain. User permissions require that the user give explicit permission before the MIDlet can access protected functions or APIs.

A MIDlet can divide its permissions into critical and optional. Critical permissions are necessary for the MIDlet to operate at all. Optional permissions are noncritical and the MIDlet can operate without them, although with reduced capabilities. User permissions can also be divided into subsets. Blanket permissions are valid for every invocation of an API by a MIDlet suite, until the suite is removed or the user changes the permission. Session permissions are valid only until the MIDlet suite terminates and must be renewed for each invocation of the suite. One-shot permissions must be confirmed by the user for every invocation of the protected API.

The new security model in MIDP 2.0 provides not only for network security, but also for protection of the device. With secure HTTP and network connections, transmitted data is encrypted to protect it from prying eyes and network scanners. Inside the device, MIDlet authorization allows sensitive data and secure capabilities to be protected from unauthorized use. Together these new capabilities provide much-enhanced protection for both the mobile device and the user's data.

User Interface

The MIDP 1.0 user interface is quite limited, in keeping with the limited capabilities of the hardware it was expected to run on as well as the wide range of devices the MIDlets must run on. MIDP 2.0 recognizes that mobile device hardware is becoming more powerful with larger screens, faster processors, and more memory. The MIDP user interface components (LCDUI) have been extended and enhanced to take advantage of this increased power.

One limitation in MIDP 1.0 is that it's not possible to specify the display positions for Items. The reason for this is that because display sizes change so much in mobile devices, the MIDP 1.0 Expert Group decided the mobile device was the best authority for deciding how to lay out content. As a result, it's left to the implementation to place things on screen as best it can, which doesn't always result in a pleasing screen display.

To solve this problem, the Form and Item classes in MIDP 2.0 have been enhanced with a set of layout directives. Items can now be assigned horizontal and vertical layout values, affecting where their containing Form will display them. Developers can also now specify that a line break should appear before or after an Item, as well as whether an Item should be shrunk or expanded to help it fit in a particular position. In addition, the appearance of a StringItem or ImageItem can be specified: they can take an appearance value of Plain, Hyperlink, or Button. Two other changes have been made to Items: they can have commands added to them and programmers can now specify minimum and preferred sizes to display the Item. The minimum size is the smallest size at which the Item can function, and the preferred size is the smallest size that allows the entire Item to be displayed (with no clipping and a minimum of wrapping).

Another limitation of the 1.0 user interface classes is that they're not extensible. Developers are limited to the items defined by the specification and can't easily create new types of user interface widgets. MIDP 2.0 solves this problem by introducing a new class, CustomItem, as an extension of the LCDUI's Item class. A CustomItem can be subclassed to create new visual and interactive elements for Forms. These subclasses are responsible for responding to user interaction events such as pointer actions or key presses. They must also define the visual appearance of their content, including sizing, rendering, colors, fonts, and graphics. (The visual appear-

esri

www.esri.com

ance of the CustomItem's label and border are handled by the implementation.) Finally, they're responsible for calling Item.notifyStateChanged() to notify listeners when their state changes.

Like other Items, CustomItems have the concept of minimum and preferred sizes that define the area needed by the entire Item. They add the concept of content size – the size of the content contained by the CustomItem (not including borders and label). The CustomItem subclass is responsible for handling events and displaying data in the area defined by the content size. The CustomItem subclass can support one or more of the user interaction modes defined by the Form on which it is displayed, but it's not required to support all possible interactions.

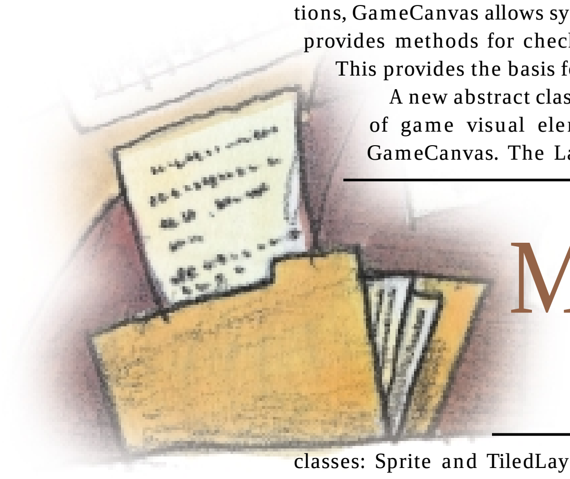
Finally, MIDP 2.0 introduces a new subclass of Item called a Spacer. This is a simple display element that's noninteractive. It allows a programmer to define its minimum height and width, making it useful for adjusting spacing between visual elements in a Form or for defining a minimum height for a row. A Spacer's label must always be null, and an application cannot add Commands to a Spacer.

Games

Games have been one of the driving applications on MIDP devices so far, but MIDP 1.0 provides little support specifically for game developers. Everything has to be done with the bare-bones MIDP 1.0 graphics functions, making game development much more difficult than it needs to be. The Expert Group that defined the MIDP 2.0 specification listened to the desires of the game developers and added classes to the specification to simplify game development and speed up game performance on most devices.

The gaming classes begin with a new subclass of Canvas called GameCanvas. In addition to the standard Canvas functions, GameCanvas allows synchronous graphics flushing and provides methods for checking the state of the game keys. This provides the basis for a game user interface.

A new abstract class called Layer allows the creation of game visual elements to be displayed on the GameCanvas. The Layer class has two defined subclasses: Sprite and TiledLayer. The TiledLayer class lets the developer create background and relatively fixed display objects. The Sprite class can be used to create animated foreground objects. Since a game may have several Layer subclasses (e.g., multiple Sprites moving in front of a TiledLayer background), a LayerManager class simplifies and automates the rendering process to display the Layers, making it easier for developers to define and maintain the foreground-to-background display order.



"The Expert Group that defined the MIDP 2.0 specification listened to the desires of the game developers and added classes to the specification"

A Sprite is a Layer subclass that can be rendered from one of several identically sized frames stored in an Image. These frames can be displayed in sequence to animate the image. The Sprite's location can be changed, and it can be made visible or invisible. It can also be flipped and rotated about the horizontal and vertical axes. Sprites can detect collisions with Images, TiledLayers, and other Sprites.

A TiledLayer is a subclass of Layer that defines a grid of cells that can be filled with a set of tile images. This allows a developer to create a game background by combining elements from the tile images, rather than by using a large Image object. The tiled images are provided in a single Image object, and the tile size is specified along with the Image. A developer can also define animated tiles, which are groups of tiles associated with a single tile image. Just changing the image in the single tile can change all the tiles in the associated group.

The LayerManager class is used to simplify the display of the various Layer subclasses that make up a game. It maintains an ordered list of Layers that defines the display order, and provides a view window that lets the developer control the size of the visible region and its position relative to the LayerManager's coordinate system. In addition to controlling the view window's position in this system, the developer can also provide an offset for its position on the physical screen to make room for game controls or status displays.

Sound

Besides the display components, a major component for many applications is sound. People like to hear as well as see things happen. In some games, music is used to heighten tension or set a mood. Sounds and music can also be used by other applications, such as music players, or for distinguishing various types of event notifications. Unfortunately, the first MIDP release could not generate sounds except for predefined alerts. The JSR118 Expert Group addressed that problem in MIDP 2.0 and expanded the range of sound possibilities.

One complicating factor for this was the JSR135 Mobile Media API, which includes sound APIs in its scope. The JSR118 team was careful to define a set of sound APIs and functions that would be compatible with JSR135 in order to maintain upward compatibility with the more complex Mobile Media API. In keeping with this goal, the Expert Group defined an API set that allowed tone generation and audio playback while remaining protocol and content-format agnostic and using a minimal amount of resources.

The resulting specification defines three main parts:

1. A Manager controls the audio resources available on the device.
2. Applications use the Manager to request Players, and to query for the audio device's capabilities.
3. A Player is responsible for actually playing the audio content, and a Control interface exposes the different controls that a Player might have.

MIDP 2.0 also defines a PlayerListener interface for receiving asynchronous events generated by Players.

These APIs can be used in two ways: an application can use the Manager and/or Player classes to generate and play a single tone or a tone sequence, or the Player class can be used to play back sampled or synthesized sound formats.

qualcomm

qualcomm.com

The specification also defines two interfaces to control volume and tone generation. The ToneControl interface enables playback of a user-defined monotonic tone sequence, and the VolumeControl interface allows an application to manipulate the audio volume of a Player.

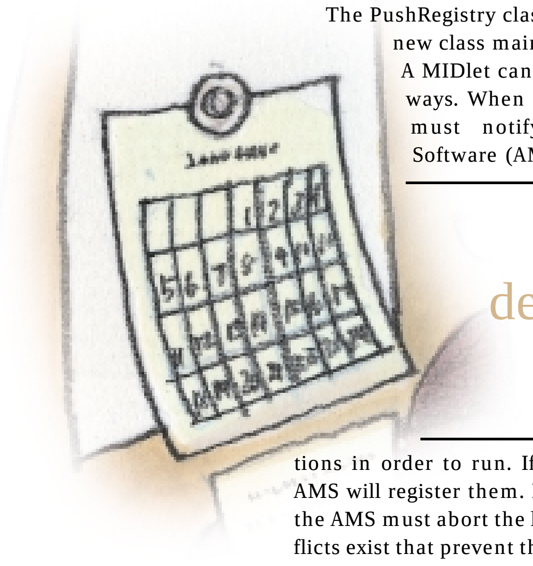
Application Downloading

Shortly after the first MIDP specification was released, a supplementary document came out describing recommended practices for Over-the-Air (OTA) provisioning of MIDlets to mobile devices. Although OTA downloading is optional for MIDP 1.0 devices, it's mandatory for MIDP 2.0. The Expert Group adopted the OTA provisioning method defined in the recommended practices document with minor changes. The MIDP 2.0 specification requires that devices be capable of discovering and downloading applications using the HTTP 1.1 protocol. Devices that communicate using the WAP June 2000 protocol must use an intermediate gateway to communicate with the HTTP provisioning server. Although HTTP provisioning is required, devices may also use other methods for downloading MIDlets, including iRDA, serial, or Bluetooth technologies.

Pushy, Pushy, Pushy

One of the great advantages of mobile devices is their ability to monitor some process (even if it's just the passage of time) and notify the user that an event has happened. The ability to receive unexpected communications and act on them is especially useful. Early J2ME devices could not do this because the first MIDP specification had no provision for receiving unexpected communication events. Their communications model was essentially that of a Web browser, reacting to user input and initiating communications only when the user wanted to.

The PushRegistry class in MIDP 2.0 changes all that. This new class maintains a list of inbound connections. A MIDlet can register a set of connections in two ways. When it is first loaded, its descriptor file must notify the Application Management Software (AMS) that it requires certain connections in order to run. If the connections are available, the AMS will register them. If the connections are not available, the AMS must abort the loading and notify the user that conflicts exist that prevent the MIDlet from being installed. Once a MIDlet is installed, it can register additional push connections using the dynamic registration methods in the PushRegistry. Once registered, the MIDlet has exclusive use of a connection.



“With MIDP 2.0, small mobile devices can become fully functional mobile data systems, enabling their users to stay connected wherever they go”

Responsibility for registered connections is shared between the AMS and the MIDlet. A MIDlet that's running is required to handle all the communications on its registered connections. When the application is destroyed, the AMS assumes the responsibility of listening for inbound connections and starting the registered MIDlet when data is received. Whatever data was received is then passed on to the MIDlet, which assumes responsibility for the connection.

Communication Options

The first version of the MIDP specification required only that HTTP connections be supported, and so defined only an HttpConnection class. Experience has shown that additional connectivity is very desirable, so MIDP 2.0 defines several optional (but highly recommended) connection interfaces. These include the UDPDatagramConnection, SocketConnection, ServerSocketConnection, and SecureConnection. None of these are required in a compliant MIDP 2.0 implementation, but they should be implemented if possible.

The UDPDatagramConnection provides applications the opportunity to use UDP Datagrams for sending and receiving messages whose delivery is not guaranteed by the underlying network stack. The SocketConnection provides stream connections to specified hosts and ports without specifying a particular protocol. ServerSocketConnections allow an application to open a stream waiting for inbound connections, and the SecureConnection interface extends the SocketConnection to provide SSL connections for data streams.

In addition to these network connections, MIDP 2.0 defines a serial communication interface, the CommConnection, which defines a logical serial port connection. The logical serial port used by this connection is defined by the underlying host platform and may not correspond to a physical RS-232 serial port.

Conclusion

The MIDP 1.0 specification was a good first attempt at defining a subset of Java that could run on very restrictive, limited-hardware devices. Experience rapidly showed that additional features were needed to improve the capabilities of Java MIDlets intended for these devices. MIDP 2.0 addresses those needs and recognizes the greater functionality of newer cell phones and other small mobile devices. Increased networking capabilities, enhanced user interface classes, and a host of other improvements provide a more robust, full-featured programming environment that enables application developers to create better applications more easily. At the same time, more powerful security features protect the user's data from inappropriate use and permit sensitive data to be exchanged safely with remote servers. With MIDP 2.0, small mobile devices can become fully functional mobile data systems, enabling their users to stay connected wherever they go.

• • •

For more information on the specification, the companies, and the people involved in developing it, please visit http://jcp.org/jsr/detail/118.jsp.

Author Bio

Roger Ritter is a programmer and developer support specialist for Motorola, Inc. He supports mobile device manufacturers who are porting Motorola's J2ME implementation to their own products.

roger.ritter@motorola.com

hit

www.hit.com

JSuite 6.0

by Infragistics Corporation

REVIEWED BY PAUL FREY *paulfrey@yahoo.com*

info

Infragistics Corporation
Windsor Corporate Park
50 Millstone Road
Building 200 - Suite 150
East Windsor, NJ 08520
Phone: 800-231-8588
Web: www.infragistics.com

Specifications
Platforms: Any platform supported by JDK 1.1 through 1.4 (there are older versions of the components included for older versions of the JDK as well)
Pricing: $795 (includes PowerChart Server Edition)
$995 with annual subscription – a year's worth of updates, upgrades and new products (includes source code)
$1,495 Enterprise Edition – product, subscription, as well as guaranteed priority support

Test Platform
Gateway GP7-600, 600 MHz Intel Pentium III processor, 80GB disk, 512MB RAM, Windows NT with Service Pack 6a

As most Java developers know, the standard GUI components provided with the Java platform are barely adequate for most applications. We've all had to extend the base Swing (JFC) components and AWT components to develop the rich user interface components that users expect today. Today's sophisticated users expect more interactive, intuitive GUI components to be available in their applications.

Swing was a good start but it lacked many of the common features that users have come to expect in a Windows world. Swing offers a good framework but the default implementations are lacking robust capabilities such as masked field entry, maximum length entry, standard popup components such as date selectors and calculator pads, and the list goes on. Of course, it should be understood that Sun was providing a base implementation that a company like Infragistics could build on and create GUI components that not only met users' expectations but surpassed them. Even those same users who once tolerated HTML form controls within a browser are now demanding richer Web-based GUI components in the form of Java applets and Sun's Java Web Start protocol. JSuite 6 from Infragistics, a company that provides cutting-edge client-side and server-side components, meets this challenge. This product contains a class of Enterprise-level, Java-based visual components for application client development, server-side components for server-side utility operations, and a broad set of client-side chart components that will satisfy the demands of today's business user community.

JSuite 6

JSuite 6 is a suite of client-side visual components and server-side data/utility components for Java-based applications or applets. The suite of components integrates easily with the most popular Java IDEs such as JBuilder, WebGain (formerly Visual Café), and VisualAge. JSuite 6 includes multiple versions of their visual components that work best with the type of application you're developing. They offer AWT, Swing, and bean-based components. Whether you're working with a Swing-based client application or an applet-based Web application, JSuite 6 will satisfy all your visual component needs. It offers components that fulfill the following visual component categories required in most sophisticated client applications:

- *Editing components* such as combobox (image/text dropdown), currency, date edit, mask edit, numeric, password, and static text
- *Date-based components* such as Calendar, DayView, WeekView, and DateEdit with dropdown calendars; advanced features include appointment scheduling with synchronization between all date-based components and DayView (with multiple lines and images per appointment)
- *Advanced table component* that provides a view to complex back-end data models and allows for easy user interaction with the underlying data model
- *Advanced tree component* that offers many utility functions that are often very burdensome for the developer, such as drag-and-drop functionality
- *Rich data-driven components* to allow easy navigation of hierarchical data structures using an Explorer-style interface or Gantt chart interface

Other powerful tools available in the JSuite 6 tool chest are:

- PowerChart, a powerful 2D/3D chart-rendering component for client applications/applets
- Utilities for common user interface operations such as color picker and customizable buttons
- Data model adapters to handle numerous back-end data needs such as JDBC, XML, text, or binary

With JSuite 6, a Java developer can get down to the business of providing a business solution quickly.

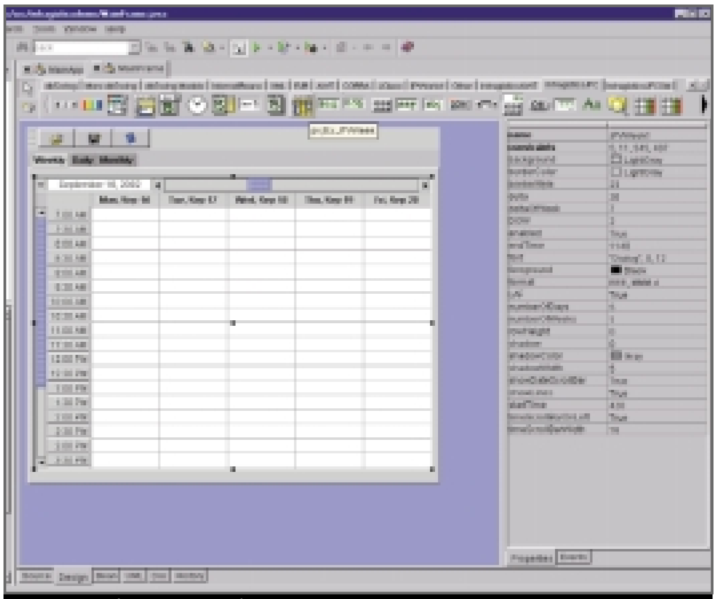


FIGURE 1 JSuite 6 components in IDE

ATTN: Developers

STEP-UP to the mic and be...

Go to http://developer.sys-con.com

HEARD!

Calling Sleek Geeks Everywhere!

Make sure you have your finger on the pulse of i-Technology...bookmark http://developer.sys-con.com today.

i-Technology
News

i-Technology
Views

i-Technology
Comment

i-Technology
Debate

SYS-CON MEDIA

JSuite 6.0 by Infragistics Corporation

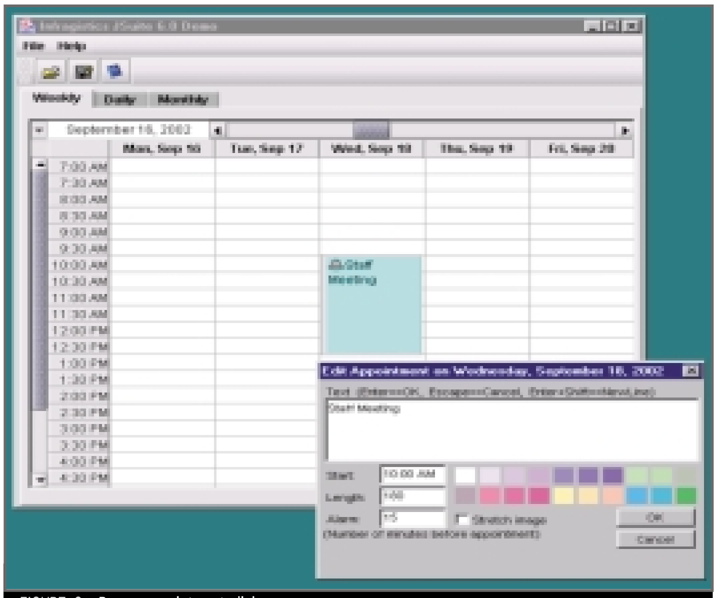

Edit Appointment on Wednesday, September 18, 2002
Text (Enter=OK, Escape=Cancel, Enter+Shift=NewLine)
Staff Meeting
Start: 10:00 AM
Length: 1:00
Alarm: 15
Stretch image
(Number of minutes before appointment)

FIGURE 2 Popup appointment dialog

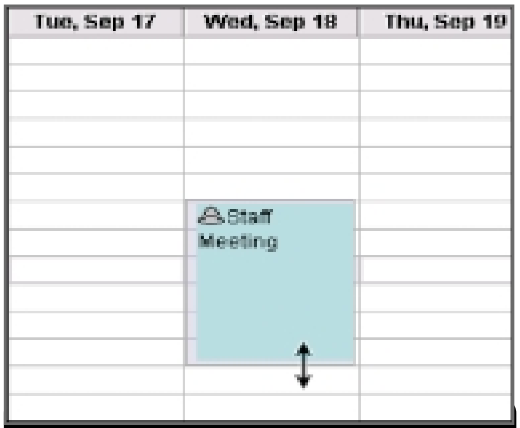


FIGURE 3 Resizing an appointment

Installing and Using JSuite 6

Infragistics offers JSuite 6 in a downloadable format. A CD key is provided to you upon successful purchase from their Web site. For those developers who wish to extend its components and need to know the internals, the source can be purchased as well. I downloaded the JSuite 6 self-extracting installation file and ran it trouble-free on a Windows NT workstation.

I opened up the JSuite Help that was created during the installation process and followed the directions for integrating the JSuite components with my preferred IDE, JBuilder. Integrating the JSuite components into JBuilder's component palette was as simple as integrating any third-party component library. Simply add a new tab on the component palette for "Infragistics" and associate this tab with the JSuite component JAR files located in the JSuite 6 installation's subdirectory, jsuite/jars. JSuite 6 icons will then be copied to JBuilder's component palette for use in the JBuilder designer.

From this point, using the components is the same as using any other Swing/AWT component in your IDE's designer. Just drag and drop the JSuite component you wish to use onto your application's panels. There are also numerous sample applications/applets source code to help you understand how to use each JSuite visual component and nonvisual component.

Using JSuite 6 from within an application was simple and intuitive. I was able to create a daily planner application in just minutes using its calendar components. The only code I had to write was the synchronization code for the components to capture the "add appointment" events. In Figure 1, the JPVWeek calendar component was dragged from the JBuilder component palette and dropped on the application's panel in the designer. The properties associated with the JSuite components are documented in the JSuite 6 Help under the "How To" section and the JavaDocs are included as well.

I found the JSuite calendar components to be user friendly. Adding an appointment was handled through the calendar components by displaying a popup appointment dialog, supplied by JSuite (see Figure 2). The user can display the appointment dialog by clicking a popup indicator, and can resize (shorten or lengthen) an appointment by positioning the mouse cursor over the bottom edge of the appointment and dragging it in either direction (see Figure 3).

JSuite 6 provides Java developers with a vast array of almost every visual component they will need when developing professional client applications. I'm unable to cover all the components in the space of this review. However, Infragistics supplied ample precompiled, ready-to-run applications that demonstrate all the visual and nonvisual components available in JSuite 6. I went through each one of these samples and found that nearly every feature demanded of me during my past Java client application projects could have been satisfied and exceeded with JSuite 6. I haven't gone into the nonvisual components in great detail, but I think many Java developers who ever wrote code using Java's JTable component with an XML or JDBC data source will appreciate JSuite's JPVTable component and its data model adapters, JPVXMLAdaptor and JPVJDBCAdaptor.

Summary

When it comes to Java client-side application development, Infragistics' JSuite 6 offers one of the most comprehensive suites of visual components available in the Java third-party visual component market. I was impressed with the Rapid Application Development (RAD) qualities the suite delivered. I would have liked to see better integration with JBuilder's IDE designer properties panel. Often I had to refer to the JavaDoc for the JSuite 6 component to determine the proper property settings. A popup editor would be helpful. In today's highly competitive business environment, an impressive front end is as important as the business-driving back end. Project managers will appreciate the return on investment (ROI) benefits of using a mature, full-featured visual component library like JSuite 6. If you are a Java developer looking to shorten your development cycle and improve your client application's usability and visual appeal, you'll want to add Infragistics' JSuite 6 to your toolkit.

JDJ Product Snapshot

Target Audience: Java developers and business analysts

Level: Beginner to advanced

Pros: Easy integration with IDEs, customizable rich set of components with intuitive built-in behavior, helpful and full-featured samples, back-end data adapters that will improve data acquisition, PowerChart components included

Cons: Need more examples in the "How To" help section (although scanning the sample code did answer many of my questions)

isavvix

www.isavvix.com

Adaptive Server Anywhere Version 8

by *i*Anywhere Solutions

REVIEWED BY Breck Carter *bcarter@risingroad.com*

info

***i*Anywhere Solutions**
Web: www.ianywhere.com
Phone: 800 801-2069

Test Platforms
Windows 95/98/Me/NT/2000/XP/CE, Netware, Compaq Tru64 Unix, IBM AIX, HP-UX, Sun Solaris, Caldera, Mandrake, Red Hat, SuSE, TurboLinux, Windows 98, and Windows 2000

Pricing
SQL Anywhere Studio 8.0 (Developer Edition) $399

The latest version of Adaptive Server Anywhere (ASA) marks a major turning point in the history of this product. Prior to version 8 the most important design goals were ease of use, small footprint, and cross-platform support, with high speed taking a back seat. This time, improved performance is the number one new feature. And the results? Mostly good, sometimes uneven, getting better fast.

Product Description

ASA is an affordable relational database management system that ships as part of the SQL Anywhere Studio package from *i*Anywhere Solutions, a subsidiary of Sybase. ASA supports all the features you expect from a modern RDBMS, including ANSI standard SQL, multiuser network connectivity, multiprocessor support, transaction commit and rollback, row-level locking, referential integrity, BLOBs, events and triggers, and stored procedures.

Installation and Setup

Installation of SQL Anywhere Studio on a Windows platform uses a straightforward InstallShield setup. A full developer's installation requires about 120MB disk space in one place under C:\Program Files\Sybase\SQL Anywhere 8. Multiple versions (5, 6, 7, 8) can coexist and even run together on the same machine. In other words, an ASA installation doesn't take over your machine or get in the way of other products, even database servers from other vendors.

New Features

There's good news and bad news, and it's all part of the same story: the query optimizer and execution engine have been completely rewritten. These are the runtime components that analyze your SQL commands and pick from among the thousands of different possible "plans" of execution. They were rewritten for three reasons: to support improvements in the database file structure, to support future SQL enhancements, and to make queries run faster.

The good news is that most queries do run faster, sometimes much faster. In a series of tests I ran against an old application, the improvements ranged from a few percentage points to 75% faster. Other people have reported queries running up to 10 times faster. Full table scans are no longer the problem they once were, sometimes running faster than index searches. Queries that need temporary tables are also faster, and indexes using wide columns such as "last_name, first_name" benefit from a new storage scheme. The bottom line is you don't have to work so hard to optimize SQL commands; the server will do it for you.

The bad news is that improvements aren't guaranteed. A rewrite brings behavior changes, and in some rare cases queries actually run slower on version 8. These "queries from hell" are being dealt with as they turn up; I know this for a fact, having reported some of my own nasty SQL.

There's a do-it-yourself fix for most of the performance disappointments, however. Just change a database option with this command in interactive SQL:

```
SET OPTION PUBLIC.Optimization_goal =
  'All-rows';
```

This tells the server to pick execution plans that favor the retrieval of entire result sets. This option didn't matter in earlier versions of ASA, but now it's critically important. The current default value "First-row" is wrong for most applications and it's going to be changed to "All-rows" in 8.0.2.

Sometimes, however, it's up to you to make a SQL statement run faster. For example, an index on "last_name, first_name" won't help a search on first name (try finding all the "Susans" in the phone book). In these situations there's nothing ASA or any other database server can do without your help, and that's where the new graphical plan display comes in.

Figure 1 shows a SELECT where two tables are being scanned sequentially. The graphical plan shows how the server handles the query, including estimated and actual runtime statistics, giving you more than enough information to help make decisions about indexes and other improvements.

The graphical plan is interactive: it lets you pick the main select and subqueries for display, and click on individual nodes in the plan to see the details in the right-hand panel or a popup box. You can even save the graphical plan in an XML file for later display without connecting to the database, a real time-saver when you're dealing with distributed data-

nary

ww.nary.com

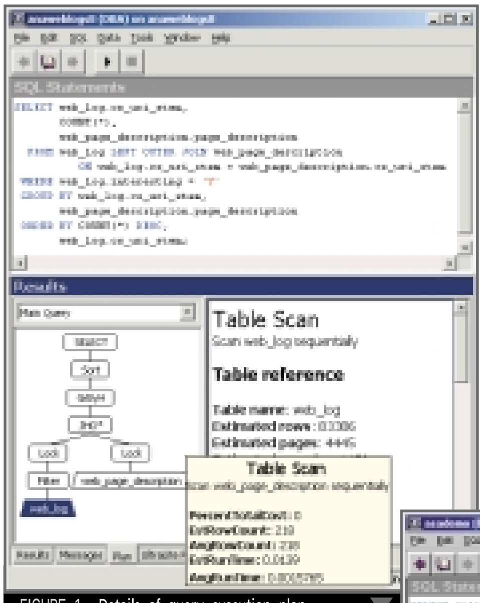


FIGURE 1 Details of query execution plan

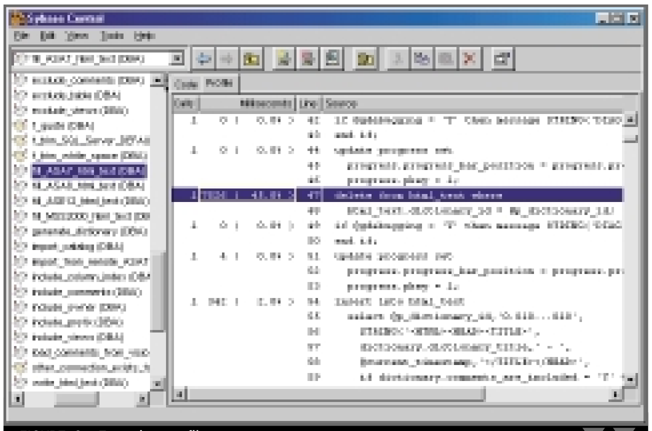


FIGURE 2 Execution profiler

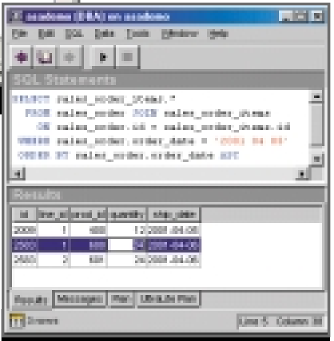


FIGURE 3 Result set editor in ISQL

bases or you need someone else's advice about optimization.

Version 8 also comes with a new execution profiler to help you find slow SQL statements. Experience shows that a few statements take up a lot of the time in most applications. Not only that, but it's almost impossible to predict which statements will be the slow ones.

The execution profiler lets you find the troublemakers inside stored procedures, even inside triggers and user-defined functions called from other statements.

Figure 2 shows an example in which 45% of the total time was spent executing a single DELETE statement. Hundreds of other statements took almost no time to execute, so no matter how efficient or inefficient they were, they didn't need attention. Just this DELETE that turned out to be unnecessary: it removed temporary data stored in a permanent table. A simple design change to use a temporary table eliminated the need for the DELETE altogether.

The speed and usability of the Java-based administrative tools Sybase Central and Interactive SQL have been greatly improved. The native C version of ISQL still ships with the product, but only the Java tools have cool features like the two new editors for queries and result sets. Figure 3 shows a join of two tables where you can edit and save changes to the database. You can insert, delete, and update rows; copy and paste to and from columns; and confirm or cancel each change – all without writing any code other than a SELECT.

Summary

The performance improvements in version 8 are moving ASA into the world of enterprise databases. ASA is inexpensive and easy to administer, as well as being developer-friendly – a viable alternative to Oracle, SQL Server, DB2, and ASE.

JDJ Product Snapshot

Target Audience: Programmers, database administrators

Level: Beginner to advanced

Pros:
- Improved performance
- Graphical plan display
- Execution profiler
- Strong database and communication encryption
- Java 2 and JDBC 2 support
- Newsgroup and customer support

Cons:
- Market perception as a small database-only product

north woods .com

SYS-CON Media Named the World's Fastest Growing Magazine Publisher

For the Third Year...

Inc. 500 2002 WINNER

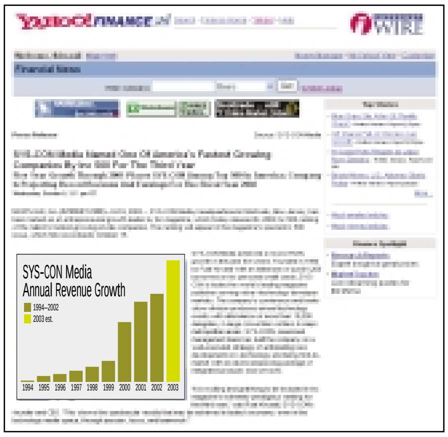
Inc. 500 2001 WINNER

Inc. 500 1999 WINNER

Deloitte & Touche Technology Fast 500 2002 WINNER

Deloitte & Touche Technology Fast 50 2002 WINNER

Deloitte & Touche Technology Fast 50 2001 WINNER



From 1998 to 2001, revenue grew at a compound annual growth rate of 72.9%.

Smart strategies can succeed even in the toughest of times...

The 2002 Inc 500 reveals a surprising resiliency within the entrepreneurial sector, where leading companies are continuing to show dramatic rates of growth despite the recession. "This is the first Inc 500 ranking to reflect the full impact of the recession," said Inc editor John Koten. "Yet these entrepreneurs are managing to confound the naysayers and move ahead despite obstacles. They are showing that smart strategies can succeed even in the toughest of times."

SYS-CON's revenue and earnings have grown dramatically since its inception in 1994. From 1998 to 2001, revenue grew at a compound annual growth rate of 72.9%. For the same period, gross margin increased at a CAGR of 75.7%. This year, as of October 2002, adjusted annual EBITDA will increase 57.7% to new record earnings. In 2003, the company projects its gross margin to increase 51.9% and the contribution is projected to increase 70.4%, which will keep SYS-CON in an impressive growth pattern for 2002, 2003 and beyond.

As a result of this impressive growth, SYS-CON Media has been recognized three times by Inc 500, twice named by Deloitte & Touche to its "Technology Fast 50" award and is expected to be named this year to Deloitte & Touche's "Technology Fast 500" award, which honors the 500 fastest growing technology companies in the United States and Canada, both among public and privately held corporations.

For more information please visit www.sys-con.com

"The Oscars of the Software Industry"

Java Developer's Journal Readers' Choice Awards Recognized:

29 Winners & 87 Finalists

Widely referred to as the "Oscars of the software industry," the *JDJ* Readers' Choice Awards program has become the most respected industry competition of its kind.

The polls were open for just under a year, from November 6, 2001, through September 23, 2002, and no fewer than 30,000 *Java Developer's Journal* readers cast their votes. The total number of award nominations also reached a new record high, with more than 180 companies nominating over 705 products in 29 different award categories. That is seven more categories than last year, and the new ones reflect emerging trends in the Java space, such as "Best Wireless Java Application" and "Best XML Tool."

Java Developer's Journal Readers' Choice Award recipients are selected through reader-submitted nominations, followed by online voting. An independent research firm manages the voting process.

BEST JAVA BOOK

Winner: *Java Message Service*
by Richard Monson-Haefel and David A. Chappell
from O'Reilly & Associates
1st Runner-Up: *J2EE Applications and BEA WebLogic Server*
by Michael Girdley, Rob Woollen, and Sandra L. Emerson
from Prentice-Hall
2nd Runner-Up: *Core J2EE Patterns: Best Practices and Design Strategies*
by Deepak Alur, Dan Malks, and John Crupi
from Prentice-Hall
3rd Runner-Up: *Java in a Nutshell* by David Flanagan
from O'Reilly & Associates

BEST DATABASE TOOL OR DRIVER

Winner: Oracle9*i* Business Components for Java
from Oracle
1st Runner-Up: XML Spy Suite 4.2
from Altova
2nd Runner-Up: Oracle9*i*AS TopLink
from Oracle
3rd Runner-Up: MM.MySQL JDBC Driver
from MySQL AB

BEST ENTERPRISE DATABASE

Winner: Oracle9*i* Database
from Oracle
1st Runner-Up: IBM DB2 Universal Database v7.2
from IBM
2nd Runner-Up: Borland JDataStore
from Borland Software Corp.
3rd Runner-Up: MySQL
from MySQL AB

BEST J2EE IDE

Winner: IBM WebSphere Studio Application Developer 4.0
from IBM
1st Runner-Up: Forte for Java/NetBeans
from Sun Microsystems
2nd Runner-Up: Oracle9*i* JDeveloper
from Oracle
3rd Runner-Up: Borland JBuilder
from Borland Software Corp.

BEST J2ME IDE

Winner: IBM VisualAge Micro Edition
from IBM
1st Runner-Up: Oracle9*i* JDeveloper
from Oracle
2nd Runner-Up: Borland JBuilder MobileSet
from Borland Software Corp.
3rd Runner-Up: CodeWarrior for Java
from Metroworks

BEST J2SE IDE

Winner: Oracle9*i* JDeveloper
from Oracle
1st Runner-Up: Borland JBuilder
from Borland Software Corp.
2nd Runner-Up: Forte for Java/NetBeans
from Sun Microsystems
3rd Runner-Up: IntelliJ IDEA
from JetBrains, Inc.

BEST JAVA APPLICATION

Winner: Oracle9*i* JDeveloper
from Oracle
1st Runner-Up: Borland JBuilder
from Borland Software Corp.
2nd Runner-Up: BEA WebLogic Server 7.0
from BEA
3rd Runner-Up: IntelliJ IDEA
from JetBrains, Inc.

BEST JAVA APPLICATION SERVER

Winner: BEA WebLogic Server 7.0
from BEA Systems
1st Runner-Up: IBM WebSphere Studio Application Server 4.0
from IBM
2nd Runner-Up: Oracle9*i* Application Server
from Oracle
3rd Runner-Up: JBoss
from jboss.org

BEST JAVA CLASS LIBRARY

Winner: The Java Collections Framework
from Sun Microsystems
1st Runner-Up: Oracle9*i* JDeveloper – BC4J BI Beans
from Oracle
2nd Runner-Up: Host Access Class Library
from IBM
3rd Runner-Up: Eclipse
from Eclipse Technology, Inc.

BEST JAVA COMPONENT

Winner: ILOG JTGO
from ILOG, Inc.
1st Runner-Up: Oracle Business Components for Java
from Oracle
2nd Runner-Up: BEA WebLogic Portal
from BEA
3rd Runner-Up: Sitraka JClass
from Sitraka

BEST JAVA DATA ACCESS TOOL

Winner: Oracle Business Components for Java
from Oracle
1st Runner-Up: XML Spy Suite 4.2
from Altova
2nd Runner-Up: Oracle9*i*AS TopLink
from Oracle
3rd Runner-Up: CA Advantage EDBC for Java Data Access
from Computer Associates

BEST JAVA DEVELOPER STUDIO

Winner: Together ControlCenter 5.5
from TogetherSoft
1st Runner-Up: IBM WebSphere Studio Application Developer 4.0
from IBM
2nd Runner-Up: Oracle9*i* JDeveloper
from Oracle
3rd Runner-Up: Borland Enterprise Studio for Java
from Borland Software Corp.

BEST JAVA EAI PLATFORM

Winner: IBM WebSphere MQ
from IBM
1st Runner-Up: Oracle9*i* Application Server
from Oracle
2nd Runner-Up: BEA WebLogic Integration
from BEA
3rd Runner-Up: SonicXQ 1.0
from Sonic Software Corp.

BEST JAVA E-BUSINESS FRAMEWORK

Winner: IBM WebSphere
from IBM
1st Runner-Up: Oracle9*i* JDeveloper with Business Components for Java
from Oracle
2nd Runner-Up: BEA WebLogic Server 7.0
from BEA
3rd Runner-Up: Borland JBuilder
from Borland Software Corp.

BEST JAVA IDE ENVIRONMENT

Winner: IBM WebSphere Studio Application Developer 4.0
from IBM
1st Runner-Up: NetBeans
from Sun Microsystems
2nd Runner-Up: Oracle9*i* JDeveloper
from Oracle
3rd Runner-Up: Borland JBuilder
from Borland Software Corp.

BEST JAVA INSTALLATION TOOL

Winner: InstallAnywhere
from Zero G Software
1st Runner-Up: InstallShield MultiPlatform
from InstallShield
2nd Runner-Up: Java Message Queue
from Sun Microsystems
3rd Runner-Up: Sitraka DeployDirector
from Sitraka

BEST JAVA MESSAGING TOOL

Winner: IBM WebSphere MQ
from IBM
1st Runner-Up: Oracle9*i* Application Server
from Oracle
2nd Runner-Up: SonicMQ 4.0
from Sonic Software Corp.
3rd Runner-Up: BEA WebLogic Server 7.0
from BEA

BEST JAVA MODELING TOOL

Winner: MagicDraw UML 5.0
from No Magic, Inc.
1st Runner-Up: Oracle9*i* JDeveloper
from Oracle
2nd Runner-Up: Rational Rose Professional J Edition
from Rational Software Corp.
3rd Runner-Up: Together ControlCenter 5.5
from TogetherSoft

BEST JAVA PROFILING TOOL

Winner: Oracle9*i* JDeveloper
from Oracle
1st Runner-Up: Metameta Debug Enterprise
from Metameta
2nd Runner-Up: IBM VisualAge Micro Edition – MicroAnalyzer
from IBM
3rd Runner-Up: Borland Optimizeit Suite 4.11
from Borland Software Corp.

BEST JAVA REPORTING TOOL

Winner: StyleReportEE
from InetSoft Technology Corp.
1st Runner-Up: Oracle9*i*AS Reports/Discoverer
from Oracle
2nd Runner-Up: Together ControlCenter 5.5
from TogetherSoft
3rd Runner-Up: JReport
from Jinfonet Software

BEST JAVA TESTING TOOL

Winner: Borland Optimizeit Suite 4.11
from Borland Software Corp.
1st Runner-Up: Oracle9*i* JDeveloper
from Oracle
2nd Runner-Up: JUnit
from Object Mentor, Inc.
3rd Runner-Up: Rational Suite TestStudio
from Rational Software Corp.

BEST WIRELESS JAVA APPLICATION

Winner: IBM WebSphere Transcoding Publisher
from IBM
1st Runner-Up: Oracle9*i* Application Server
from Oracle
2nd Runner-Up: *i*Anywhere m-Business Studio
from *i*Anywhere Solutions, a Sybase company
3rd Runner-Up: COOL:Plex/Websydian
from Computer Associates

BEST XML TOOL

Winner: XML Spy Suite 4.2
from Altova
1st Runner-Up: IBM WebSphere Studio Application Developer 4.0

from IBM
2nd Runner-Up: Oracle9*i* XDK
from Oracle
3rd Runner-Up: Borland JBuilder
from Borland Software Corp.

MOST INNOVATIVE JAVA PRODUCT

Winner: Together ControlCenter 5.5
from TogetherSoft
1st Runner-Up: IBM WebSphere Studio Application Developer 4.0
from IBM
2nd Runner-Up: Oracle9*i* Application Server with Advanced Clustering
from Oracle
3rd Runner-Up: Borland JBuilder
from Borland Software Corp.

BEST JAVA VIRTUAL MACHINE

Winner: Oracle9*i* JVM
from Oracle
1st Runner-Up: The J9 Virtual Machine
from IBM/Object Technology International
2nd Runner-Up: Java HotSpot Performance Engine
from Sun Microsystems
3rd Runner-Up: BEA WebLogic JRockit Server Side VM
from BEA

BEST TEAM DEVELOPMENT TOOL

Winner: MagicDraw UML 5.0 Teamwork Server
from No Magic, Inc.
1st Runner-Up: Oracle9*i* SCM
from Oracle
2nd Runner-Up: Borland JBuilder
from Borland Software Corp.
3rd Runner-Up: Rational ClearCase
from Rational Software Corp.

BEST MOBILE DATABASE

Winner: PointBase Micro 4.1
from PointBase
1st Runner-Up: Oracle9*i* Lite
from Oracle
2nd Runner-Up: Cloudscape
from IBM
3rd Runner-Up: Borland JDataStore
from Borland Software Corp.

BEST JAVA WEB SERVICES DEVELOPMENT TOOLKIT

Winner: IBM Web Services Toolkit
from IBM
1st Runner-Up: Oracle9*i* JDeveloper
from Oracle
2nd Runner-Up: Borland Web Services Kit for Java
from Borland Software Corp.
3rd Runner-Up: SonicXQ 1.0
from Sonic Software Corp.

BEST JAVA TRAINING PROGRAM

Winner: Java Productivity with JBuilder
from Borland Software Corp.
1st Runner-Up: Java BluePrints for Wireless Program
from Sun Microsystems
2nd Runner-Up: JCertify 5.0
from EnterpriseDeveloper.com
3rd Runner-Up: Struts Fast Track Public Training
from baseBeans Engineering

JAVANEWS>

QUALCOMM Selects TIBCO's Integration Solution

(Palo Alto, CA) – TIBCO Software, Inc., has announced that QUALCOMM, Inc., a provider of digital wireless communications products and services, has selected TIBCO's Business Integration Solution for enterprise-wide integration.

QUALCOMM believes it will be able to reduce integration expenses and TIBCO's solution will enable it to specify integrations and gain access at various levels of the same messaging technology.
www.qualcomm.com
www.tibco.com

Communix Launches Provisioning Solution for J2EE Application Servers

(London) – Communix Limited, a value-added reseller of e-business infrastructure products and services, has launched AppSynergy, a solution that reduces the cost and time it takes to procure, install, and configure best-of-breed and leading-vendor J2EE application server software on branded workstation and server-class computers for development, test, and production environments.

AppSynergy also offers product choice and flexibility. Among the products available are IBM WebSphere, BEA WebLogic, Oracle Databases and Application Server (9*i*AS), Borland JBuilder and Enterprise Server (BES), TogetherSoft ControlCenter 6.0, and Macromedia JRun 4.
www.communix.com

IBM's New Tools Complete Web Services Plunge

IBM has unveiled IBM WebSphere Studio v5 (formerly VisualAge development platform), an important milestone in the company's strategy to help its customer base use Java and Web services to unify their legacy infrastructure. It also cements tool support for key technologies and standards, including J2EE 1.3, the Apache Struts framework, the Eclipse 2.0 tool framework, and the latest versions of all the Web services standards.
www.ibm.com

Borland Signs Agreement to Acquire Starbase

(Scotts Valley and Santa Ana, CA) – Borland Software Corporation and Starbase Corporation, a provider of software configuration management technology, have signed a definitive agreement in which Borland will acquire Starbase in an all-cash tender offer for an aggregate purchase price of approximately $24 million or $2.75 per share. Starbase's products will complement and expand Borland's suite of products.

The acquisition has been approved by the boards of directors of Borland and Starbase and is subject to customary closing conditions.
www.borland.com
www.starbase.com

Macromedia Delivers Flash on Java

(San Francisco) – Macromedia has unveiled a new version of its Flash Remoting technology – which lets its Flash technology run as a server-driven application – that will run on any Java application server. In addition to providing server streaming of Flash content, it also provides links to back-end business logic and data stored on enterprise servers.

Flash Remoting supports Java objects and JavaBeans, and J2EE resources including Java classes, EJBs, and JMX MBeans. The server is a pure Java implementation and can be deployed on both J2EE and Java application servers.
www.macromedia.com

Sun Delivers Secure Mobile Access with Sun ONE Portal Server

(Santa Clara, CA) – Sun Microsystems, Inc., has unveiled the Sun ONE Portal Server, Secure Remote Access 6 product. It provides an integrated, out-of-box alternative. Using a standard Web browser, users can securely access their portal (personalized and aggregated applications, data, Web services, and legacy information) from any location, such as an Internet café, airport, or home.
www.sun.com

SYS-CON MEDIA AWARDED TOP HONORS BY INC 500 AND DELOITTE & TOUCHE

(Montvale, NJ) – SYS-CON Media has been named one of the fastest growing 500 technology companies in North America by Deloitte & Touche in its 2002 Technology Fast 500. The announcement came one week after SYS-CON was named one of the nation's fastest-growing private companies by Inc 500 for the third time.

SYS-CON Media is widely recognized in the *i*-technology and magazine publishing industries as the world's leading publisher of print magazines, electronic newsletters, and accompanying Web portals. The company has further solidified its dominant role in the *i*-technology space with the 2000 launch of an events business with trade shows, conferences, and education seminars.

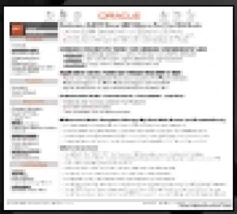
SYS-CON Media achieved a record 752% growth in the past five years. The company's revenue and earnings have grown dramatically since its inception in 1994. From 1998 to 2001, revenue grew at a compounded annual growth rate of 72.9%.

In 2003, the company projects its gross margin to increase 51.9%, and the contribution is projected to increase 70.4%, which will keep SYS-CON in an impressive growth pattern for 2002, 2003, and beyond.
www.sys-con.com

ORACLE DELIVERS PERSONALIZED JAVA DEVELOPMENT TOOL

(Redwood Shores, CA) – Oracle Corp. has announced the delivery of the latest version of its Java Integrated Development Environment (IDE) – Oracle9*i* JDeveloper version 9.0.3. This new release of Oracle9*i* JDeveloper features four major areas of enhancement. Developers can download the new tool directly from Oracle Technology Network (OTN) to take immediate advantage of full support for the latest J2EE 1.3 specifications; enhanced Web services support; built-in integration with open-source tools; and the new MyJDeveloper Extension Manager – a unique feature that allows developers to personalize the development environment to meet their exact project needs.

With full support for open-standards development on any operating system, Oracle9*i* JDeveloper provides complete development life-cycle support for Java developers creating J2EE applications and Web services. It offers built-in features for optimizing the performance of Java applications while providing a single IDE for Java, XML, and SQL; business intelligence; UML modeling; and J2EE Web services.



www.oracle.com

Exclusive: Excerpts from JavaDevelopersJournal.com

JDJ MEETS... JEREMY ALLAIRE

by JDJ News Desk

ALAN WILLIAMSON interviewed Jeremy Allaire, CTO, Macromedia, Inc., to get the rundown on how Java developers can take advantage of JRun4, ColdFusion MX, and Flash Remoting.

JDJ: *With CFMX in the field for nearly three months now, what is the adoption rate like?*

Allaire: We're seeing a lot of excitement about ColdFusion MX. For existing users, the most attractive features of ColdFusion MX have been the new component model and the ability to easily work with XML and Web services. They're also excited about the possibility of running their ColdFusion applications on a Java application server. For Macromedia, however, the best part of this release is the interest we're getting from people who haven't previously used ColdFusion. A lot of shops that have standardized on J2EE are now looking at ColdFusion as a great way to do rapid application development on their Java server, and the Macromedia Flash development community is excited about the way Macromedia Flash Remoting allows them to easily connect their applications to a database.

JDJ: *CFML has a reputation in the field for being too lightweight for anything serious. How have you addressed this issue with the Java community?*

Allaire: There are two important sides to this question, and we think ColdFusion MX addresses both. On the development side, we've not only preserved the productivity and ease of use of our scripting environment, but also introduced a powerful new component model that makes it much easier to do structured development in ColdFusion. As a result, you can easily implement advanced design patterns like MVC using the same high-level scripting syntax that makes ColdFusion a great tool for rapid application development. Moreover, we don't restrict you from using other languages. If there is something you would rather develop in Java, that's fine. You can easily reuse that code from within ColdFusion – by calling the object directly, sharing data with a servlet, or importing it as a tag library.

The other side of this question is in deployment, and this is where ColdFusion MX represents a major change. By rearchitecting ColdFusion MX so that it can run on a Java server, we're enabling developers who are using ColdFusion to take advantage of the performance and reliability of the J2EE platform. For example, if you deploy your ColdFusion application on IBM WebSphere, it will run just like any other application on WebSphere. It can run in multiple instances; it can use the ultrafast IBM virtual machine; and it can take advantage of specialized WebSphere features such as load balancing, database connection pooling, vertical scaling, and legacy integration.

• • •

To read more of this interview, go to www.sys-con.com/java.

OBJECTFOCUS INC.
Leader in Enterprise Software Staffing

The Right Person
The Right Job
The First Time

XML / XSL J2EE
Senior++ Engineers
Web Services Swing EJB
WebLogic / WebSphere Architects

Looking for Enterprise Software Talent?

Looking for Enterprise Software Jobs?

Our core focus is enterprise class architects and senior developers with distributed computing and OO backgrounds, who understand how to build and deploy scalable, high-performance and robust systems through the full life cycle.

We've worked with over 700 engineers, filling critical positions in Fortune 1000 and emerging growth companies since 1995.

ObjectFocus...We're technologists too!

phone 408.973.8200 *fax* 408.873.8090 *www.ObjectFocus.com*

CONTACT US TODAY

Inadequate Project Managers

I enjoyed Jason Bell's editorial "The 84% Rule (Vol. 7, issue 9) and wanted to add an important point.

In my experience, planning a software project is not a static activity – it can't be done in advance with any degree of accuracy. Some details can only be discovered once a project is in motion, and traditional planning methods (and tools) don't account for that. It's similar to compiling code…some errors are found at compile time, others at runtime.

A PM's typical response to risk is to add more time to the estimate. But, as you said, estimates are wildly inaccurate. The better solution is to account for risk (or "confidence") as a separate item in a project plan, rather than just lumping it under the "time" column.

FYI: I gauge a project manager by how well he or she understands the time-cost-features triangle. With one exception, every project manager I've ever worked under has been inadequate.

Chris Freyer
christopher.freyer@bcbsfl.com

I do agree with you that planning software projects is a "nonstatic activity," but what I do find is that managers don't seem to realize that if a change is made to a requirement it has to be managed. Most changes do happen once a project is in motion, I completely agree. As developers and PMs, we are constantly being asked for a deadline just so marketing can do their thing and the sales guys can do their thing. What usually happens is that marketing and sales then sell something completely different than what is on the spec!

One day we'll get a manager who understands that people only work a certain amount of hours in a day, and a PM who understands what PMs are for. They have a knack for thinking they are programmers!

Jason Bell
jasonbell@sys-con.com

Bingo Card Skill Sets

In Alan Williamson's editorial "Tale of Two Camps" (Vol. 7, issue 9), the assertion that "At the end of the day we are software engineers, designed to solve problems," begs the question: Why does the industry insist on bingo card skill sets? All I've heard for the past five years is how companies want to check off certain skills, and continue to ask for people with improbable and impossible skills (Java, C++, Perl, VB…and RPG!) Is the "industry" changing from this bingo card mentality (skills + years of experience) to asking for professionals with OO design and implementation skills who are language-agnostic?

Scott McMahan
scott@skwc.com

Simplicity and Productivity

I've been using this IDE since it was released ("IntelliJ IDEA 3.0" by Duane Fields [Vol. 7, issue 9]). I've tried all of the big names: JBuilder, Cafe, VisualAge, Forte, etc., and this is by far the best. It has the simplicity of a basic text editor with the productivity of a Java IDE. I even use it to teach my Java classes because it doesn't "get in the way" of learning Java.

Dave Ford
dford@smart-soft.com

The Clear Winner Is SWT

I couldn't agree more with Alan Williamson's editorial "Swing Is Swinging Java out of the Desktop" (Vol. 7, issue 10). While it may be possible for Swing gurus who write Java bytecodes in their sleep to create Swing-based programs that are both reasonably fast and have an "almost native" look and feel, SWT gives you all of this for *free*.

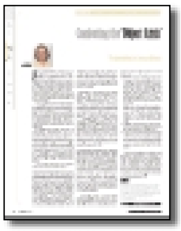
Although others will argue that it is possible for talented Swing gurus to create Swing applications with good performance and a native look/feel (and I wouldn't dispute this), they miss the point. At issue is not what a highly experienced Swing architect can accomplish but what your average C# programmer can do.

In this battle, SWT is the clear winner.

David Orme
daveo@asc-iseries.com

Think Objects

I agree with Jacquie Barker("Combating the 'Object Crisis'" [Vol. 7, issue 9]). However, I wonder if there's another factor. New Java technologies appear to be springing up all the time. I'm mostly a core Java developer, but stuff like EJB (before 2.0), servlets, JSF, and struts, don't seem to be very object-oriented. Maybe getting those new Java developers to think objects will be more difficult than you think.



Tony Weddle
tony@maestro.demon.co.uk

Real Problem — Wrong Argument

Jason Weiss, in his editorial "Is Complexity Hurting Java?" (Vol. 7, issue 10), is attacking a real problem but his arguments are completely wrong. The page count of the J2EE spec doesn't tell you anything about the learning curve for the average applications developer. For the most part, the specs provide a formal unambiguous definition of what the contractual responsibilities of an application server implementer are. In fact, you can learn JSP in a day. VB and PowerBuilder are heavily based on SQL right?

Counting acronyms doesn't prove anything. All you need to know to use JDBC, for example, can be said in about 100 lines of code and explained in two hours. JNDI is about as hard to learn as java.util.Map.

The real problem on the server side is EJB. It's just flawed. It doesn't solve the right problems and it's unnecessarily complicated and inflexible. Let's just throw it out and be done with it.

Alexander Jerusalem
ajeru@vknn.org

ASK: Making Server-Side Development Easy

WRITTEN BY ENRIQUE PÉREZ GIL

I do believe the center of Java development is the programmer who is creating object-oriented Java code. But how do you achieve this when developing Web applications? In the Internet scenario the client and server sides are disconnected: the front end is shown to the user miles away from where the real code that's reacting to it is executed. Even worse, the visual code is different from the code that executes the logic. How much time does a Java Web application developer dedicate to tasks that differ from real application logic? Forty percent? Thirty percent? And what about maintenance?

ASK is a front-end application server that provides the necessary support for numerous requests. This is achieved with a server-side server's topology. Within this architecture it's possible to set up as many ASK servers as you need, indicating which applications every ASK server should support, and also which applications run in which servers. All the architecture is described in one XML file.

ASK is built on the premise that the application logic should be 100% Java, 100% object-oriented, and 0% HTTP/HTML. To achieve that goal, ASK implements a Java server-side graphical component model that melts the client and server sides. All the HTTP complexities, like session management and navigation, are internally managed by the ASK front-end application server, thus Web application development is actually visual Java components and events handlers development.

The most typical problems that I've seen in Web application projects have been those that are related to team management trying to fit the application aspect tasks (designers, stylists) with the application logic ones (Java developers). It's hard to mix these two worlds with an efficient methodology. Using ASK helps solve this problem. The development process is as follows: Java developers develop the application and spend 99% of their time making it work (Visual Logic, DDBB, EJBs), without worrying about color, font, etc. Once it's ready, and that means really working, the stylist team arrives to set the skin. Since responsibilities are clearly separated, communication between the teams is minimal, reducing the overall development and maintenance time.

Most Web applications need to be executed in a secure, scalable, performance-efficient, and, hopefully, low-cost environment. ASK was developed from scratch taking all that into consideration but adding another requirement: simplicity.

A recursive question I get asked is: "What happens if the server crashes?" The answer is: ASK is applications fault-tolerant. It offers an applications crash-and-recovery service that's plugged into those servers where needed. Also, it's not active for all the applications; you decide which type of application should use it independently.

Performance is an issue, and this has been checked from the start of the development, conditioning many low-level design decisions. An ASK server is a simple thing; it doesn't need a lot of resources or to execute complex or heavy processes to handle applications. It's lightweight – the ASK code is compressed to 300KB. The amount of time an ASK server spends executing a call to your application is almost the same amount of time it takes your code to execute. You may have performance problems if the number of concurrent requests increases, as in any other situation, but then the solution in ASK is simple: set up another ASK server (with a low-cost but powerful machine, if possible).

ASK is an open-source (lesser GPL) tool that proposes a new, simple, and powerful way to develop Web applications, where simplicity and openness are a must.

There are literally thousands of open-source projects in the works at the moment, many of them based on very innovative and exciting technologies…many of them not! We want to shine some light on the more innovative and smaller projects – projects that don't necessarily have a large body or company behind them to give them the exposure they deserve.

Spotlight on Open Source is brought to you by fellow developers. We want to hear about the projects that you think are notable and perform a great service for the world of Java. Please e-mail me with your suggestions for future **Spotlight** features at alan@sys-con.com.

Resources

- www.openode.org
- http://sourceforge.net/projects/openode
- http://enrique.blog-city.com

epgawt@terra.es

AUTHOR BIO

Enrique Pérez Gil is the director of the e-business area for Virtual Desk SL, based in Madrid, Spain (www.virtualdesk.es). He has 12 years of development experience, six of them developing Web-based Java projects for banking, logistic, and e-commerce companies. Enrique holds a BS in computer science from Madrid Polytechnic University.

Playing Four Square

How hiring managers view engineers

WRITTEN BY
BILL BALOGLU &
BILLY PALMIERI

You're a senior Java engineer who's been working with J2EE on enterprise systems software and applications. You've got a résumé that reads like a who's who and what's what of current technologies.

But when you apply for a job that looks like a perfect fit for your skills, you get a rejection letter. Or even worse, it seems as if your résumé was sucked into a black hole and you never hear anything back at all.

What's going on, you wonder. Why don't these people realize how perfect I am for this position?

Potential employers look at a résumé in a way that's fundamentally different than you might expect. And a lot of it has to do with the type of work – and the kind of company – that's listed on your résumé.

To understand how people in the industry perceive different types of engineering experience, imagine a box made up of four squares. From the top left, number the squares clockwise, as 1, 2, 3, and 4. These numbers have nothing to do with seniority levels; they're merely a way to visualize four different types of work (see Figure 1).

Boxes 1 and 4 represent the work done at product companies. Boxes 2 and 3 represent the work done at IT organizations. The engineers who work at level 1 build commercial applications that sit on top of the core infrastructures built by engineers at level 4.

When we refer to product companies, we're talking about companies that sell enterprise-wide products to other companies. These include Oracle, Siebel, Arriba, and BEA. Using Oracle as an example, the level 4 engineers are the ones who built the Oracle core platform. The level 1 engineers build applications that run on top of it.

Engineers who build commercial applications must face a special set of problems and challenges. Those who have worked in this environment have experience with building a product that's going to market.

Engineers who work at levels 2 and 3 work for IT organizations, doing similar yet different kinds of work. Examples of IT organizations include the IT divisions of large companies such as Charles Schwab, E*TRADE, or DHL.

The engineers at level 2 build applications, for the company's internal use, that sit on top of the core infrastructure built by engineers at level 3. While these may be powerful, complex systems and applications, there's a fundamental difference between this kind of work and building a product that's going out the door.

In the eyes of those within the industry, it's like looking at two automobile engineers who are both building Fords. One engineer has been building racing cars, the other has been building consumer cars. That's not to say that one type of work is better than the other, but each product demands different skills and experience from the engineer who builds it.

By the same token, an actor who's had a long career in situation comedies is going to have a hard time selling himself to someone who's casting a production of *Hamlet*. He might be a great Shakespearean actor, but it's going to take some special strategies to sell himself for the role.

Many people fit into more than one of these imaginary boxes, and there's no reason why a skilled engineer couldn't make the transition from one type of work to another. It's a matter of perception on the part of potential employers, and how well you can address those perceptions when applying for the job.

The first step is to realize that based on your résumé, you are being perceived not only by your skills, but by the type of work you've done in the past. The second step is to specifically address potential concerns in a cover letter or within the résumé itself.

Spamming out the same résumé for many different types of positions is the most common mistake made by job-hunting engineers. It takes more time to tailor your résumé and cover letter to each position, but in our experience, it's the only way to catch the serious attention of hiring managers.

An unfortunate fact of life in the hiring world is that most résumés aren't carefully read, they're scanned. Since we're currently in a buyer's market, it's even more critical that you sell yourself carefully and thoughtfully, taking as many issues into account as possible.

By recognizing in which of these four squares a hiring manager might perceive you, you stand a much better chance of positioning yourself to cross over into one of the other squares.

It's ultimately about depth and breadth of experience. Once you've built racing cars, Aerostars, and Escorts, you should be able to get any job building any kind of car you want.

jdjcolumn@objectfocus.com

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 position was at Renaissance Worldwide, where he held several senior management positions in the firm's Silicon Valley operations.

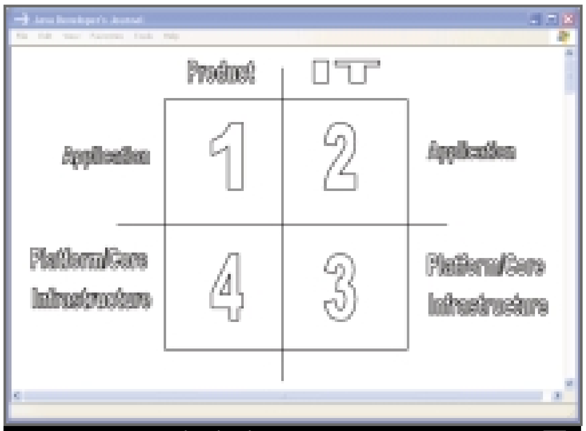


FIGURE 1 Four areas of engineering

JAVA DEVELOPER'S JOURNAL ADVERTISERINDEX

ADVERTISER	URL	PHONE	PAGE
AccelTree	www.acceltree.com		30
Altova	www.altova.com		27
AltoWeb	www.altoweb.com		55
Apple Computer, Inc.	www.apple.com/xserve		4-5
BEA	www.dev2dev.bea.com/useworkshop		9
Borland Software Corp.	www.borland.com/new/javasolutions/92122.html		41
Bowstreet	www.bowstreet.com/easy		17
Crystal Decisions	www.crystaldecisions.com/cr9/002/	888-333-6007	35
Engenuity Technologies	www.jloox.com	800-684-5669	45
ESRI	www.esri.com/arcims	888-289-5084	57
HiT Software	www.hitsw.com	408-345-4001	61
IBM	ibm.com/developerWorks/linux/cd		11
IBM	ibm.com/websphere/ebaydev		33
InetSoft Technology Corp.	www.inetsoft.com/jdj	888-216-2353	53
Infragistics, Inc.	www.infragistics.com	800-231-8588	14-15
iSavvix	www.isavvix.com	1-866-iSavvix	65
JBoss Group	www.jboss.org		28
Jinfonet	www.jinfonet.com/jdj11.htm	301-838-5560	31
Macromedia	www.macromedia.com/go/jrun4jdj		37
Motorola	www.motorola.com/developers/wireless		6
/n software inc.	www.nsoftware.com		43
n-ary	www.javaSOS.com		67
New Atlanta Communications	www.newatlanta.com		47
Northwoods Software	www.nwoods.com/go/	800-434-9820	68
Oracle Corp.	oracle.com/javafaster	800-633-0546	21
Our World Live	www.ourworldlive.com		Cover III
Parasoft Corporation	www.parasoft.com/jdj11	888-305-0041	39
Precise Software	www.precise.com/jdj	800-310-4777	29
QUALCOMM Incorporated	www.qualcomm.com/brew		49, 59
Rational Software	www.rational.com/offer/javacd2		13
Sitraka	www.sitraka.com/jclass/jdj	800-663-4723	25
Sitraka	www.sitraka.com/jprobe/jdj	800-663-4723	51
Sitraka	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		23
SYS-CON Media	www.sys-con.com		69
SYS-CON Subscription Offer	www.sys-con.com/suboffer.cfm		71
WebAppCabaret	www.webappcabaret.com/jdj2.jsp		38
Web Services Edge 2003	www.sys-con.com	201-802-3069	73
Zero G	www.zerog.com	415-512-7771	3

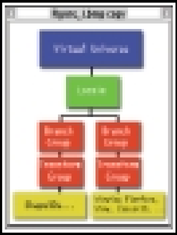
What's in the next issue of *JDJ*?

SEARCH-ENABLE YOUR APPLICATION WITH LUCENE

Lucene is an open-source search framework from Apache's Jakarta project. This article shows you how to use Lucene to build a search solution for your application. Although the 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.

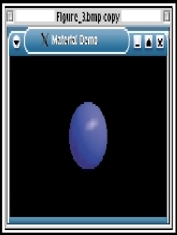
PLUG IN YOUR COMMAND PROCESSOR NOW AND START SAVING MONEY!

The Command Processor 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, as well as kick the tires on the Regular Expression package included in the 1.4 JDK.



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.

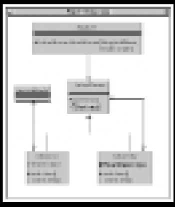


JREPORT BY JINFONET

Enterprise reporting products create output reports from databases and other data sources (such as XML documents). The enterprise reporting market can be divided into two basic categories – query and reporting. Query tools are generally geared toward end users and are designed for interactive analysis and drill-down (from summary data into detailed reports). Enterprise reports such as JReport output data into report formats such as hard-copy printouts and HTML pages.

ROUGH SEAS, SINKING SHIPS, AND LIFEBOATS

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.



Choose to Foos

WRITTEN BY
BLAIR WYMAN

I'm at the COMMON computer conference this week in Denver, and writing a Cubist Thread is about the last thing I should be doing at the moment.

Since I'll be presenting sessions throughout the week, I should really be reviewing them to make sure I have my message straight or at least make sure I don't goof up too badly. I'm still trying to live down that episode when I credited Sir Francis Bacon, of all people, with the invention of Java. (Sorry, Dr. Gosling!) It's just that kind of faux pas that can affect a person's technical credibility!

Actually, I have the sessions down pretty well, I guess, though I won't know until I'm standing in front of the audience. It's rarely the case that any Java presentation can remain unchanged from conference to conference; so many things change so quickly that bringing the information up to date is always a concern. While I'll be talking about our iSeries JVM, another project I work on called Remote AWT, and some JNI topics, I think the most exciting subject is the new content in version 1.4 of Java: the new I/O, assertions, regular expressions, and logging, to name a few – very cool new stuff in an already cool language.

I arrived in Denver last night, and it was my only "free" night here, so naturally (instead of reviewing my sessions or getting a good night's sleep) I had to seek out some local foosball. I happened to find a wonderful Web site dealing exclusively with Colorado foosball that listed a half-dozen or so likely foosing venues in downtown Denver. After a nice dinner (on the company, naturally) I took my little hand-scrawled map and struck out on my own to find just the right table.

Since my foosball **Thread** last year ("Ahh, Youth..." [***JDJ***, Vol. 6, issue 12]), I've been getting a bit more serious about the game. I have my own table now and have been practicing on a fairly regular basis, so naturally I like to think I'm getting pretty good. After all, my foos team was runner-up at the site-wide foosball tournament at work, so I figured I must be nearly unstoppable.

Oh, I knew I might run into some talented foosers, but I fully expected to be among the better players. I even envisioned returning to my hotel a wealthy foos king, having been showered with the undying admiration of the entire Denver foosing community. "Look out Denver foosers!" I thought. "I'll just go downtown and give these local foosers a taste of truly sophisticated Minnesota foosball."

Yeah, right. Either (1) there are a lot of talented foosers in this town and a few of them happened to stop by the table at which I chose to play, or (2) there are only a few talented foosers and they *all* happened to show up. Either way, I'll be looking for the shards of my shattered foosball ego for the rest of the week.

Many of my colleagues at work are taking to the game with a lot of zeal. Last year, I was on the team that won the site-wide foosball tournament, albeit barely, but there frankly wasn't a lot of competition. Only a couple of teams were in any position to seriously compete for the title. This year, it was apparent that lots of people have been practicing; any number of teams could have taken the coveted foosball title, complete with T-shirt and all-important bragging rights.

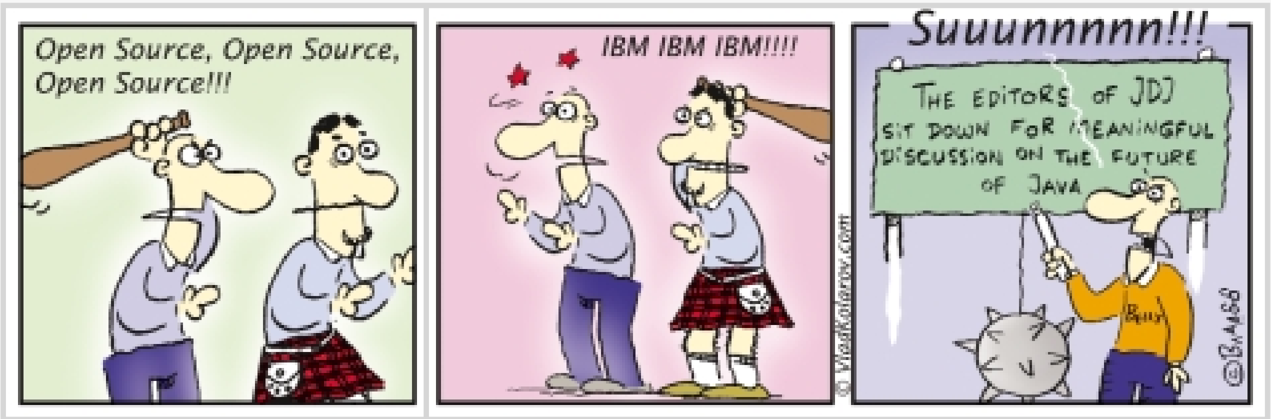
Of course, I've been doing some foosing "on the town" back home, as well, but the results have been mixed. (The eternal question: Is it "success" to beat someone so badly that they throw a cue ball at you? Yes, it really happened.) With the onset of Old Man Winter in Minnesota, I'm hoping the extracurricular foosing opportunities increase (though I might have to borrow my son's football helmet if we go back to that place).

What if programming language dominance were as simple as foosball dominance? "I happen to know a little language called Java that'll kick...." (Just look out for flying cue balls!)

blair@blairwyman.com

AUTHOR BIO
Blair Wyman is a software engineer working for IBM in Rochester, Minnesota, home of the IBM iSeries.

Java Dudes


Open Source, Open Source, Open Source!!!
IBM IBM IBM!!!!
Suuunnnnn!!!
THE EDITORS OF JDJ SIT DOWN FOR MEANINGFUL DISCUSSION ON THE FUTURE OF JAVA
© WladKolarow.com
©BAMBS

our world live

ourworldlive.com

sitraka

www.sitraka.com