SYS-CON

VOTE FOR *i*-TECHNOLOGY'S TOP 20 PAGE 3

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

JDJ

JANUARY 2005 VOLUME:10 ISSUE:1

Java Technology for the Wireless Industry

PLUS...

- SOA Solutions with J2EE
- Translation-Based Integration

SEE BROCHURE INSIDE!

web services EDGE conference & expo

PAGES 51-60
February 15-17, 2005
Hynes Convention Center
Boston, MA

- Join Over 3,000 Developers
- App Server Shoot-Out!
- FREE J2EE Tutorial
- JDJ Partner Pavilion
- Seminars and Case Studies

$1,000,000 Software Giveaway

RETAILERS PLEASE DISPLAY
UNTIL MARCH 31, 2005

$5.99US $6.99CAN

02

0 09281 01751 6

THE WORLD'S LEADING *i*-TECHNOLOGY MAGAZINE WWW.SYS-CON.COM/JDJ

Your potential. Our passion.™

Microsoft®

Feel free to think big.

With Visual Studio® .NET 2003, you write less code than with Microsoft® Visual Basic® 6.0, so you can turn that big idea into reality faster than you ever thought possible. You get a new IDE that leverages your existing skills in a tool that gives you more robust code, improved IntelliSense® and better deployment. All of which means you're more productive, and more ready to take on your biggest ideas. Find out more at msdn.microsoft.com/visual

Microsoft® Visual Studio®

i-Technology's All-Time Top 100?

Jeremy Geelan

No sooner had we begun our reader-driven quest for the Twenty Top Software People in the World than – by popular acclaim, as they say – we're extending the field of candidates... from 40 to over 100. The question remains: If these are the Top 100 Software People in the World, who are the Top Twenty?

Here are the new additions to the list. (To view the complete list go to http://sys-con.com/java.) A whole new round of voting begins this month online and will stay open 'till January 20. Have at it!

- *Gene Amdahl:* Implementer in the '60s of a milestone in computer technology: the concept of compatibility between systems
- *Marc Andreessen:* Pioneer of Mosaic, the first browser to navigate the WWW; co-founder of Netscape
- *John Vincent Atanasoff:* Inventor of an electronic computer in the late 1930s, not for fun or glory but because he had problems for it to solve
- *Charles Babbage:* Lucasian Professor of Mathematics at Cambridge in 1828; inventor of the "calculating machine"
- *John Backus:* Inventor (with IBM) of FORTRAN (FORmula TRANslator), in 1956
- *Ralph Baer:* "The man who invented video games" (Pong)
- *Kent Beck:* Creator of JUnit and pioneer of eXtreme Programming (XP)
- *Bob Bemer:* One of the developers of COBOL and the ASCII naming standard for IBM (1960s)
- *D J Bernstein:* Author of qmail
- *Fred Brooks:* Co-creator of OS/390, helping to change the way we think about software development
- *Luca Cardelli:* Implementer of the first compiler for ML (the most popular typed functional language) and one of the earliest direct-manipulation user-interface editors
- *Vincent Cerf:* "The Father of the Internet," co-inventor of the first Internetworking Protocol, TCP
- *Brad Cox:* Father of Objective-C
- *Alonzo Church:* Co-creator of the "Church-Turing Thesis"
- *Alistair Cockburn:* Helped craft the Agile Development Manifesto
- *Edgar (Ted) Codd:* "Father of Relational Databases," inventor of SQL, and creator of RDBMS systems
- *Larry Constantine:* Inventor of data flow diagrams; presented first paper on concepts of structured design in 1968
- *Ole-Johan Dahl:* Developer (with Kristen Nygaard) of SIMULA, the first object-oriented programming language
- *Tom DeMarco:* A principal of the computer systems think tank, Atlantic Systems Guild
- *Theo de Raadt:* Founder of the OpenBSD and OpenSSH projects
- *Edsger W. Dijkstra:* One of the moving forces behind the acceptance of computer programming as a scientific discipline; developer of the first compilers
- *Brendan Eich:* Inventor of JavaScript; chief architect of the Mozilla Project
- *Robert Elz:* University of Melbourne, Department of Computer Science
- *Doug Englebart:* Father of the Mouse; devised the Open Hyperdocument System; interactive computing's founding pioneer
- *Richard P. Feynman:* Legendary physicist and teacher, teacher of Caltech course (1983–86) called "Potentialities and Limitations of Computing Machines"
- *Bill Gates:* Chief software architect (and lord high chief of everything else) of "the world's #1 company" (Hoovers.com)
- *Adele Goldberg:* Developer of SmallTalk, along with Alan Kay; wrote much of the documentation
- *Andy Hertzfield:* Eazel developer and Macintosh forefather
- *Grace Murray Hopper:* Developer of the first compiled high-level programming language, COBOL
- *Jordan Hubbard:* One of the creators of FreeBSD; currently a manager of Apple's Darwin project
- *Jean D. Ichbiah:* Principal designer, Ada language (1977)

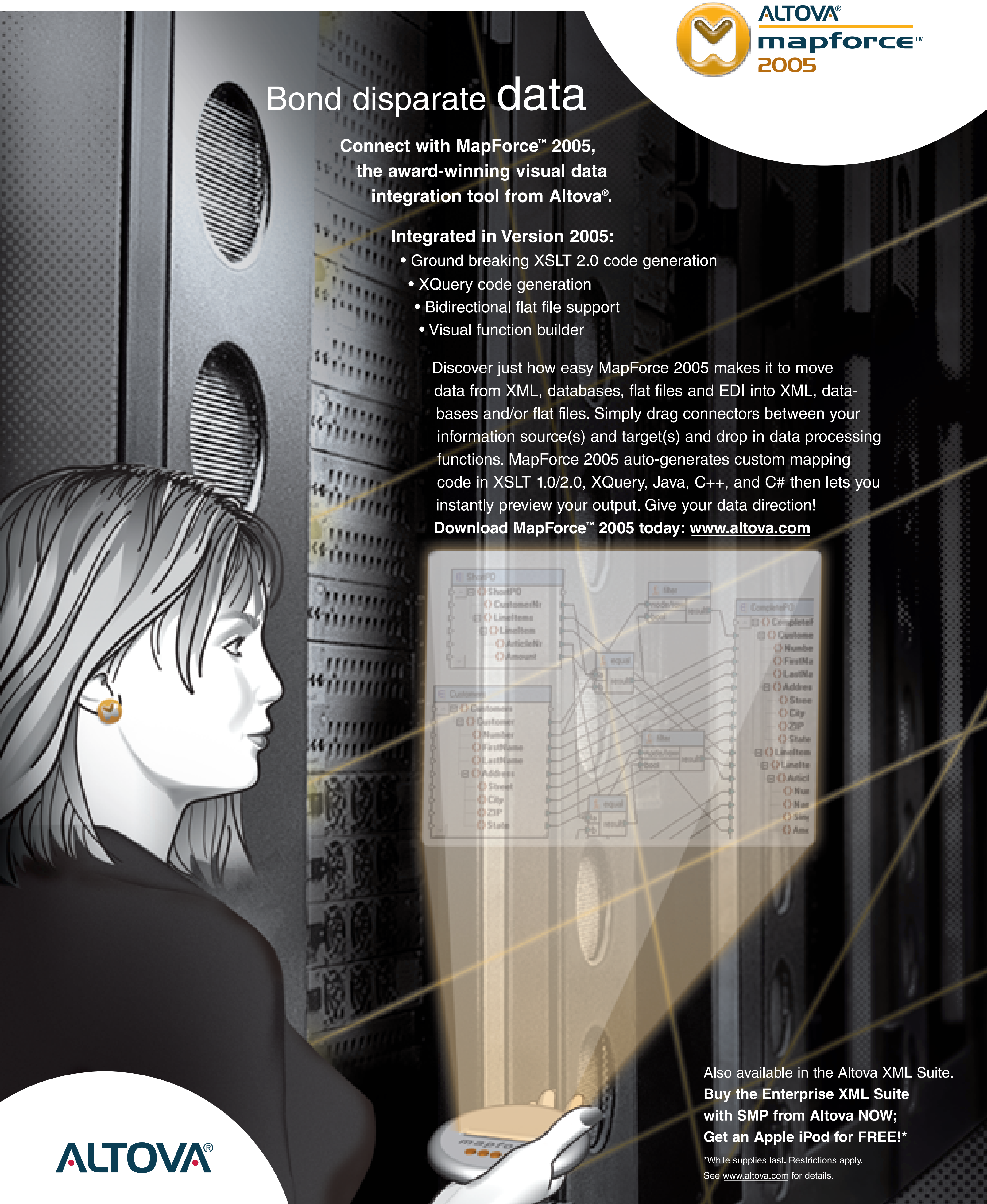
–continued on page 35

Jeremy Geelan is group publisher of SYS-CON Media and is responsible for the development of new titles and technology portals for the firm. He regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas.

jeremy@sys-con.com

Altova® MapForce™ 2005 – The premier XML/DB/flat file/EDI data integration tool.

ALTOVA® mapforce™ 2005



Bond disparate data

Connect with MapForce™ 2005, the award-winning visual data integration tool from Altova®.

Integrated in Version 2005:

- Ground breaking XSLT 2.0 code generation
- XQuery code generation
- Bidirectional flat file support
- Visual function builder

Discover just how easy MapForce 2005 makes it to move data from XML, databases, flat files and EDI into XML, databases and/or flat files. Simply drag connectors between your information source(s) and target(s) and drop in data processing functions. MapForce 2005 auto-generates custom mapping code in XSLT 1.0/2.0, XQuery, Java, C++, and C# then lets you instantly preview your output. Give your data direction!

Download MapForce™ 2005 today: www.altova.com

Also available in the Altova XML Suite.

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

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

ALTOVA®

JANUARY 2005 VOLUME:10 ISSUE:1

JDJ contents

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

J2ME: Has Its Time Finally Arrived?

Eric Giguère

Having written about Java 2 Micro Edition (J2ME) programming for almost five years now, I've been frustrated by the slow adoption of the J2ME platform, as have many of the early devotees. Those of us who saw Sun demonstrate Java running on Palm OS back in 1999 were hoping that the wireless Java revolution was just around the corner. The introduction of J2ME generated a lot of hype and excitement (and confusion) in the Java development community. Publishers rushed books to market, virtual machines were announced, and device manufacturers started incorporating Java technology into their plans. But the corner we wanted to turn was in fact a long way off. I think we're finally approaching it, though.

Writing a J2ME application isn't really that hard. J2ME is, after all, just Java, albeit with a few restrictions. The development process is the same except for a couple of additional steps. If you're an experienced Java developer, you can write J2ME applications. The plethora of acronyms will probably confuse you at first, but there's nothing radical to learn – just new application programming interfaces (APIs). If there's one thing J2ME is good at, it's defining new APIs!

No, the hardest part of J2ME development is getting your application to work within the constraints imposed by real devices. Note that here we're mostly talking about mobile phones – the primary target for J2ME applications today, even though J2ME is about more than mobile phones. Unlike simulators, devices have memory and processor speed limitations that can cripple you in unexpected ways. You may be a good Java coder, but how much of an application can you cram into 64K (or less) of memory? What's worse, all the useful classes you'd expect to find in a modern J2SE implementation are mostly missing. As you can imagine, code size reduction tools and techniques are always hot topics of discussion among J2ME developers.

Another problem with J2ME development is that buggy Java environments have been deployed to various devices. Developers are constantly tripping over bugs with the Java virtual machine and the Java application management subsystem. The manufacturers release firmware updates with bug fixes, but how realistic is it to expect the average consumer to know enough to upgrade their phone? Instead they'll try to run an application, see the application crash or hang, and give up. It's a no-win situation for the developer.

We may be turning the corner. It's been long enough now for J2ME standards to mature enough to include some important features and capabilities that were previously missing. Manufacturers are working on releasing higher-quality implementations of those standards. Minimum standards for what a mobile phone should include in terms of J2ME capabilities have also been defined in another standard known as Java Technology for the Wireless Industry (JTWI). These are all simplifying life for J2ME developers.

Just as important, a critical mass of devices is now out in the field. Again, we're talking mobile phones here – PDAs running Palm OS or Pocket PC just aren't a viable market for J2ME development as of yet, because they have alternative programming tools and languages and generally more open platforms. The only exception to this would be the BlackBerry wireless handheld, which is actually a J2ME-based device.

With most of the new handsets being produced supporting either JTWI or else its key component – version 2 of the Mobile Information Device Profile (MIDP) – developers have a more consistent and capable platform to use for application development. There are still problems that need to be addressed with application testing and deployment, but I think J2ME's time to shine is almost here. Finally!

Eric Giguère is a software developer with *i*Anywhere Solutions, where he works on mobile technologies. Eric is the author of the first-ever book on Java 2 Micro Edition programming as well as numerous articles on wireless and handheld programming, including most of the J2ME Tech Tips published by Sun Microsystems. For more information about Eric and J2ME, visit his Web site at www.ericgiguere.com.

ericgiguere@ericgiguere.com

SYS-CON MEDIA

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

Advertising
Senior Vice President, Sales and Marketing:
Carmen Gonzalez *carmen@sys-con.com*
Vice President, Sales and Marketing:
Miles Silverman *miles@sys-con.com*
Advertising Sales Director:
Robyn Forma *robyn@sys-con.com*
National Sales and Marketing Manager:
Dennis Leavey *dennis@sys-con.com*
Advertising Sales Manager:
Megan Mussa *megan@sys-con.com*
Associate Sales Managers:
Kristin Kuhnle *kristin@sys-con.com*
Dorothy Gil *dorothy@sys-con.com*
Kim Hughes *kim@sys-con.com*

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

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

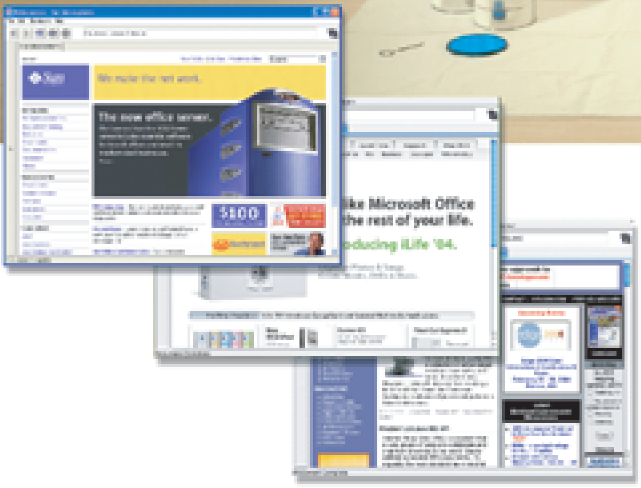
Web Services
Information Systems Consultant:
Robert Diamond *robert@sys-con.com*
Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Matthew Pollotta *matthew@sys-con.com*

Accounting
Financial Analyst:
Joan LaRose *joan@sys-con.com*
Accounts Payable:
Betty White *betty@sys-con.com*
Accounts Receivable:
Steve Michelin *smichelin@sys-con.com*

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

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

WebRenderer™



Standards compliant embeddable web browser component

WebRenderer is a cutting edge embeddable Java™ web content rendering component that provides Java applications and applets with a fast, standards compliant HTML and multimedia rendering engine. WebRenderer provides a feature-packed API including complete browser control, a full array of events, JavaScript interface, DOM access, document history and more.

Why WebRenderer?

- Standards Support (HTML 4.01, CSS 1 & 2, SSL, JavaScript, XSL, XSLT etc.)
- Exceptionally Fast Rendering
- Predictable Rendering
- Scalability (deploy in Applications or Applets)
- Security (based on industry standard components)
- Stability and Robustness

Embed WebRenderer to provide your Java' application with standards compliant web content rendering support.

To download a 30 day trial of WebRenderer visit
www.webrenderer.com

JadeLiquid™ Software

Michael Juntao Yuan
Contributing Editor

From Smart Phones to EJB 3

In the past six months, I had the opportunity to work with two leading firms in the Java world – Nokia and JBoss. Being the world's largest J2ME device vendor and most popular J2EE server developer, respectively, Nokia and JBoss come from the two ends of the Java technology spectrum, which traditionally have little to do with each other. Yet, I have learned that their visions about the future of the Java technology are strikingly similar. The core of this vision is to leverage "lightweight middleware" to deliver software services. Indeed, from Nokia Communicator phones to Eclipse-rich client applications on the desktop to EJB 3.0 application servers, lightweight middleware might be the biggest story across all Java domains in year 2005. As developers, you definitely need to know about it.

The focus on middleware in a J2ME editorial might come as a surprise to many readers. Shouldn't the hotspots in J2ME be mobile 3D games, Bluetooth P2P applications, and Internet-based mobile business forms applications? Well, yes, those are important areas of J2ME applications. However, as a platform, J2ME succeeds on the service delivery model it enables. A large part of the wireless industry is built upon service delivery. For instance, the operator-controlled mobile content delivery process enables the mobile phone ringtone download business to grow many times larger than the entire PC music business.

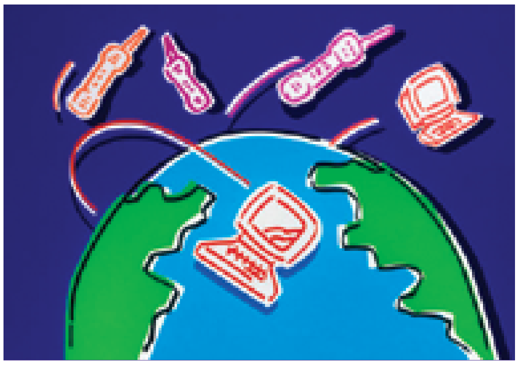
The importance of service delivery is clearly illustrated by the different fortunes of two J2ME products in the market place. MIDP (Mobile Information Device Profile) is phenomenally successful on mobile phones despite its numerous limitations; but the Personal Profile, a fully featured Java 2 environment for devices, is a market failure on PDA and handheld computers. Many people blamed Sun for failing to release a Personal Profile reference implementation on Pocket PC or Palm devices.

The real question here is: Why didn't PDA vendors rush to support J2ME Personal Profile as mobile phone vendors did for MIDP? MIDP's secret for success is that it enables developers and wireless operators to easily deliver valuable software services to the device users. The MIDP runtime environment manages the application life cycle, updates, and interaction with the outside world (e.g., the SMS message–triggered push registry). In other words, the MIDP follows a "middleware" model to deploy business components (a.k.a. the MIDlets) in managed containers. The J2ME Personal Profile environment does not provide such features and that is what has to change to revive J2ME on high-end mobile devices.

The proposed next generation architecture of J2ME (JSRs 232 and 249, led by Nokia, Motorola, and Vodafone) envisions advanced middleware services on high-end mobile devices. It focuses on remote provisioning of services, including applications, library APIs, and even system software updates, using the OSGi (Open Service Gateway initiative) micro-container on devices. This "client-side middleware" concept already has a proven track record. For instance, IBM Pervasive Computing has been selling OSGi-powered PDAs to enterprise customers for several years. In the desktop world, the popular Eclipse IDE framework uses an embedded OSGi microcontainer to run all plug-ins and rich client applications.

On the J2EE server side, while the value of middleware and service delivery have been well accepted, the existing middleware application models (e.g., EJB 2.1) are too complex for most developers and too heavyweight for most applications. Lightweight middleware is a key design concept behind the next generation EJB application servers (i.e., EJB 3.0). You can download and play with JBoss's EJB 3.0 preview software today to see what it's like. Besides EJB 3.0, it would be easy to implement OSGi services on top of the JBoss lightweight microkernel. In theory, we would then be able to deploy the same services to J2EE servers, Eclipse desktops, and J2ME devices. After years of separate development and fragmentation, J2ME and J2EE might eventually converge and unite under the vision of "lightweight middleware."

This is great news for all Java developers. Are you ready for it?



"Why didn't PDA vendors rush to support J2ME Personal Profile as mobile phone vendors did for MIDP?"

Dr. Michael Juntao Yuan is a member of *JDJ*'s editorial board. He is an expert in end-to-end mobile software solutions. He has published more than 30 articles in leading technical and academic journals in the past three years and provided consulting services to some of the biggest information technology firms around the world. Michael is the author of two books: *Enterprise J2ME* (Prentice Hall, 2003) and *Developing Scalable Nokia Series 40 Applications* (Addison-Wesley, 2004). He received a PhD from the University of Texas at Austin.

michael.yuan@sys-con.com

JANUARY
BUZZ

Jtest®

Combining AEP Practices for Effective Error Prevention

As I mentioned in last month's Buzz, the AEP practices of coding standards and unit testing should be considered two sides of the same coin. Each practice can uncover problems that the other cannot. Consequently, to identify and prevent a wide range of software problems, it's important to perform *both* coding standard analysis and unit testing, not just one or the other.

This month, I'd like to elaborate on how combining specific AEP practices is the most effective way to identify and prevent specific errors. For example, let's consider resource or memory leaks. Application-level runtime error detection and/or profiling can identify leaks, but typically not until late in the development process, when correcting the leak is typically time-consuming, difficult, and expensive. With the appropriate technologies, leaks could be detected at the unit level. However, unless the development team is required to close all resources that are opened within a unit, a leak found at the unit level might not indicate a true application-level leak.

One effective solution to this dilemma is to combine coding standard enforcement, unit-level leak detection, and application-level leak detection/profiling. First, implement a coding standard that requires that the same class or function which allocates a resource must also deallocate that resource. By checking compliance with this coding standard, you can ensure that the team's developers write code in a way that prevents leaks. If this coding standard does not make sense in all situations, individual exceptions can be documented and suppressed. You can then use unit-level leak detection to dynamically verify that leaks do not occur within the unit. After all units are developed in this manner, you can use application-level leak detection and/or profiling to verify whether leaks occur at the application level. When the team works in this manner, significantly fewer leaks reach the application level, where they are more time-consuming, difficult, and expensive to fix.

The Parasoft AEP Methodology is constructed to combine multiple practices in this supportive manner. It encourages development teams to determine which types of errors they need to prevent, then combine practices to create a "leak-proof" system for identifying and preventing those types of errors. For details about this methodology, visit **www.parasoft.com** .

– ***Adam Kolawa,*** *Ph.D.*
Chairman/CEO of Parasoft

Deliver better Java applications in less time, with fewer resources.

Parasoft® Jtest® makes Java unit testing and coding standards analysis fast and painless.

With just a click, you can verify that your Java applications are robust, high quality and secure. Jtest automatically verifies compliance to hundreds of coding rules while automatically generating and executing JUnit test cases — creating harnesses, stubs and inputs.

See how Jtest can enhance your Java development efforts...
Visit our Parasoft Jtest Resource Center for the latest information — including white papers, webinars, presentations, and more.

For Downloads go to www.parasoft.com/jtest_JDJ
Email: jtest_JDJ@parasoft.com Call: 888-305-0041 x2174

PARASOFT®
We make software work.®

Features	Benefits	Platforms	Contact Info
• Full Eclipse integration • Quick Fix automatically corrects errors • Automatically generates JUnit test cases • Customizable testing and reporting	• Makes error prevention feasible and painless, which brings tremendous quality improvements, cost savings, and productivity increases • Makes unit testing and coding standard compliance feasible and painless • Encourages the team to collaborate on error prevention	• Windows 2000/XP • Linux • Solaris	Parasoft Corporation 101 E. Huntington Dr., 2nd Flr. Monrovia, CA 91016 **www.parasoft.com**

Writing Classpath-less J2EE Clients...

. . . using a simple technique

by Bahar Limaye

Developing lightweight J2EE client applications that interoperate with multiple application servers can be difficult to do. Having to include an application server–specific JAR file along with a thin client application can significantly increase the size of the deployed application and make it too big to be practical.

In addition, client-side execution is tightly coupled using proprietary app-server extensions. This is especially true when mixing and matching multiple J2EE vendors together (such as a Tomcat servlet container communicating with a WebLogic EJB tier). Standard protocols, such as RMI/IIOP, are supposed to alleviate this problem. However, there is still no standard API or library that can be used across application servers without sacrificing native features, such as:

- Clustering and high availability (including load balancing and failover)
- Native communication protocols for optimal performance, such as socket multiplexing and efficient serialization (as opposed to standard RMI/IIOP)
- Transaction management (including distributed transactions)
- Security (JAAS, SSL transports)
- Interoperability with previous and future versions of the application server
- Lack of support for other J2EE features such as JDBC, and JMX
- Specific JDK version requirements

This article presents a simple technique using URL classloading and reflection to build portable, ultra-thin Java client applications without adding anything in the classpath, while preserving all of the application servers' native features. (The source code for this article can be downloaded from www.sys-con.com/java/sourcec.cfm.) Specifically, this article demonstrates how to:

- Use the java.net.URLClassLoader to obtain Java resources from an HTTP connection to achieve dynamic class loading and automatic software distribution while maintaining a zero client footprint.
- Invoke methods reflectively with the URLClassLoader and understand that the code is dynamically bound at runtime versus compile time.
- Create high-bandwidth and small-memory footprint JVM applications deployed as thick-client applications and utilize this technique to obtain J2EE, XML, rich client libraries, and business rules from the server.

Listing 1 is an example of an ultra-thin J2EE client application invoking an HelloWorld EJB. It obtains an InitialContext from JNDI, looks up a home interface, creates the remote object, and invokes a method. Notice that there are no Java imports to J2EE APIs or dependent classes (HelloWorld objects) for compilation and execution. A URLClassLoader is utilized to retrieve the necessary class files via HTTP on demand. All of the application servers' functionality, such as the internal communication mechanism, security, transaction management, etc., are automatically downloaded and executed. This also applies to the user-defined application code, such as the HelloWorld client stubs and home interfaces.

High-Level Overview

One of the greatest features of the Java platform is the ability to dynamically download and execute Java code from a set of URLs to a Java Virtual Machine (JVM). The result is that a remote system can run a program, for example, a thin J2EE client, which has never been installed on its disk. Using the java.net.URLClassLoader and reflection APIs, any JVM can download any Java resource, including RMI stubs, which enables the execution of method calls on a remote server using the server system's resources.

How It Works

The ThinClient.java program does not contain any imports other than the core J2SE APIs. It uses a URLClassLoader to load its Java resources (see Figure 1). The URL is a servlet (see Listing 2) that's deployed on an application server as part of an .ear file.

1. The application server binds the HelloWorld EJB into JNDI. The ClassLoader servlet and Web application (see Listing 2) is deployed on the server and includes the HelloWorld_ejb.jar as part of the classpath. When resources are accessed from this servlet, it will serve classes from its .war, EJB classes, .ear, and System classpath. (Listings 2–5 can be downloaded from www.sys-con.com/java/sourcec.cfm.)
2. The thin J2EE client application uses the URL ClassLoader to load Java resources. The particular example in Listing 1 loads all classes using this classloader.

```
// set the context class loader to a URL
ClassLoader
ClassLoader classLoader;
Thread.currentThread().setContextClass-
Loader(
    classLoader = new URLClassLoader(
        new URL[] {
            new

URL("http://localhost:7001/classloader/
ClassLoaderServlet/")
        }
    )
);
```

3. The thin J2EE client application uses reflection to instantiate a hashtable to set up the InitialContext and

Bahar Limaye is a system architect at The College Board. He has extensive experience building distributed object-oriented systems.

blimaye@collegeboard.org

performs a JNDI lookup operation reflectively. The URLClassLoader is used to download the application server's required classes for the lookup operation. The HelloWorld stubs are returned and executed on the thin client (see Listing 3).
4. The thin J2EE client performs method calls via reflection to the server (see Listing 4).

Listing 2 contains the servlet code that loads the Java resources. In this particular example, the servlet is colocated with the application in the same J2EE container.

Java resources are accessed using the following syntax:

```
http://<servername>:<port>/classloader/
ClassLoaderServlet/test/Example.class
```

The extra path information:

```
/test/Example.class
```

is a hypothetical class called test.Example in the system classpath or part of the .war or .ear files.

Alternative Technologies

A client can access an application service several ways. For example, a client may use a proprietary protocol, RMI/IIOP, JMS, or Web services. Several advantages and disadvantages exist with each of these communication technologies, and choosing the most "optimal" client application design can be challenging.

Interoperability is one factor that comes to mind when choosing a communication mechanism. Both thin J2EE clients and Web services fit well in this model. Web services give clients the ability to interoperate with almost any type of system and application, regardless of the platform on which the system or application runs. The thin J2EE client approach is limited to Java, but allows conversational state, asynchrony, transactions, security, and high-availability performance.

Web services use HTTP as the transport protocol, enabling clients to operate with systems through firewalls. The service's WSDL document enables clients and services that use very different technologies to map and convert their respective data objects. For services and clients that are based on JAX-RPC, the JAX-RPC runtime handles this mapping transparently.

Listing 5 illustrates how to create a dynamic Web service using WSDL. Notice that there are only J2EE imports. It's very similar to the ThinClient.java example, in that the operation and service definitions are dynamically obtained.

The thin J2EE client approach is faster than Web services, since it uses native communication protocols while making remote calls over the network. However, it's better suited for clients operating in intranet environments where there is a greater degree of control over the client's deployment and the J2EE server. A downside for this environment is passing through firewalls. Furthermore, this approach provides clients with secure access to the application business logic while at the same time taking care of the details of the client and server communication and marshalling and unmarshalling parameters.

Java Message Service (JMS) is another means for thin J2EE clients to communicate with server applications. It provides a means for asynchronous communication. Applications using JMS are better suited to a setting that is behind a firewall, since messaging systems generally don't work well on the Internet (often, messaging systems are not even exposed on the Internet).

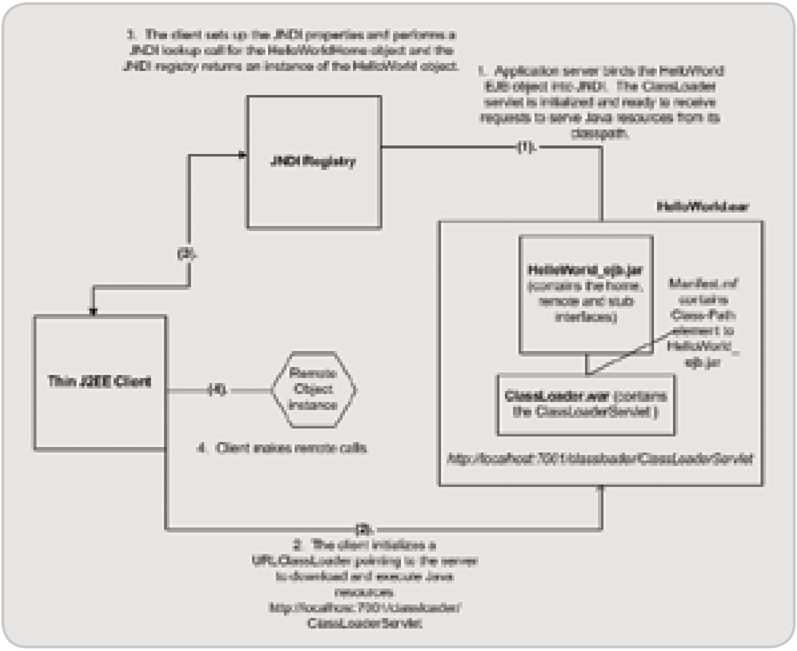

3. The client sets up the JNDI properties and performs a JNDI lookup call for the HelloWorldHome object and the JNDI registry returns an instance of the HelloWorld object.
1. Application server binds the HelloWorld EJB object into JNDI. The ClassLoader servlet is initialized and ready to receive requests to serve Java resources from its classpath.
JNDI Registry
(1).
(3).
HelloWorld.ear
HelloWorld_ejb.jar (contains the home, remote and stub interfaces)
Manifest.mf contains Class-Path element to HelloWorld_ejb.jar
Thin J2EE Client
(4).
Remote Object Instance
ClassLoader.war (contains the ClassLoaderServlet)
4. Client makes remote calls.
http://localhost:7001/classloader/ClassLoaderServlet
(2).
2. The client initializes a URLClassLoader pointing to the server to download and execute Java resources.
http://localhost:7001/classloader/ClassLoaderServlet

Figure 1 ThinClient.java program

Using Thin Clients with the WebLogic Server

Check your J2EE application server documentation to see if it comes with a resource to serve objects to the client. For example, the WebLogic server contains a built-in servlet, called ClasspathServlet, to serve Java resources in a distributed manner. The servlet code in Listing 2 is not required, since this feature comes out-of-the-box in WebLogic. The URL syntax is as follows:

```
http://<servername>:<port>/bea_wls_inter-
nal/classes
```

The bea_wls_internal is a default Web application that is enabled by default. The /classes part of the URL is registered to the internal weblogic.servlet.ClasspathServlet.

If the URL is set to /bea_wls_internal/classes, the classes required by the thin J2EE client should be available in the system classpath of the server.

If the URL is set to /bea_wls_internal/classes/DefaultWebApp@DefaultWebApp, the classes required by the thin J2EE client should be in the applications/DefaultWebApp/WEB-INF/classes directory or in the system classpath.

Versions prior to WebLogic 7 must register the ClasspathServlet and it's accessed via the /classes URL.

For this to operate successfully, you must add the following parameter to the WebLogic start script:

```
-Dweblogic.servlet.ClasspathServlet.
disableStrictCheck=true
```

This relaxes the restriction of the downloadable file type. Hence, any Java resource can be served from this servlet.

Java Web Start

Java Web Start (JSR 000056) is a built-in J2SE software-distribution mechanism that downloads and executes Java software from a HTTP server. Typically, it's used from a browser and works well with the Java plug-in. The Java Network Language Protocol (JNLP) is the underlying communication mechanism used by Java Web Start. Alternatively, Java Web Start can be invoked from the command line using the javaws utility.

The techniques presented in this article are not meant as a replacement for Java Web Start. They simply demonstrate how to create lightweight clients in your own programs when you need to selectively download and execute application code.

Summary

As service-oriented architectures re-emerge, it becomes crucial to have a decoupled and extensible client layer that can accommodate change and dynamic behavior. Constant API changes and backward compatibility problems can occur during concurrent development cycles. For an architecture that consists of all Java components (both client and server), Web services may not be the best choice, since it can cause performance degradation (overhead of XML parsing), deployment problems (load balancing/failover), and security issues.

Utilizing the native application server features, such as clustering, transactions, and security, with a loosely coupled implementation, architectures can focus on integration versus environmental problems. Constant API changes and backward compatibility problems can occur during concurrent development cycles. Building a very thin client layer can facilitate dynamic code changes and provide an adaptable and extensible architecture.

Resources

- *Tutorial: Getting Started Using RMI-IIOP:* http://java.sun.com/j2se/1.4.2/docs/guide/rmi-iiop/tutorial.html
- *Java Web Start FAQ:* http://java.sun.com/products/javawebstart/faq.html#55http://e-docs.bea.com/wls/docs81/rmi_iiop/rmiiiop2.html#DevelopingThinClient
- *java.net: Class URLClassLoader:* http://java.sun.com/j2se/1.4.2/docs/api/java/net/URLClassLoader.html

Listing 1

```
package examples;

import java.net.*;
import java.util.*;
import java.lang.reflect.*;


public class ThinClient {

    public static void main(String[] args) {
        try
        {

            // set the context class loader to a URL ClassLoader
            ClassLoader classLoader;
            Thread.currentThread().setContextClassLoader(
                classLoader = new URLClassLoader(
                    new URL[] {
                        new

URL("http://localhost:7001/classloader/ClassLoaderServlet/")
                    }
                )
            );

            // setup the InitialContext information
            Hashtable env = new Hashtable();
            env.put("java.naming.factory.initial",

"weblogic.jndi.WLInitialContextFactory");
            env.put("java.naming.provider.url", "t3://local-
host:7001");

            // equivalent of ctx = new InitialContext();
            Class ctxClass = classLoader.loadClass("javax.naming.
InitialContext");
            Constructor ctxConstructor = ctxClass.
getConstructor(new Class[]

{Class.forName("java.util.Hashtable")});
            Object ctx = ctxConstructor.newInstance(new Object[]
{ env });
            System.out.println("Context is: " + ctx);

            // equivalent of home = ctx.lookup("HelloWorldHome");
            Method method_lookup = ctx.getClass().getDeclaredMeth
od("lookup", new

Class[] { Class.forName("java.lang.String") });
            Object home = method_lookup.invoke(ctx, new Object[]
{ "HelloWorldHome"

});
            System.out.println("Home is: " + home);

            // equivalent of remote = home.create();
            Method method_create = home.getClass().getDeclaredMet
hod("create",

null);
            Object remote = method_create.invoke(home, null);

            // get sayHello and remove methods
            Method method_sayHello  = remote.getClass().getDe-
claredMethod(
                "sayHello",
                new Class[] { Class.forName("java.lang.String")
});

            Method method_remove = remote.getClass().getDeclared-
Method(
                "remove",
                null);


   // equivalent of calling remote.sayHello("Test");
            method_sayHello.invoke(remote, new Object[] { "Test"
});


            // remove the remote object
            System.out.println("Removing the Hello World Bean");
            method_remove.invoke(remote, null);

        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Nothing beats our racks Absolutely nothing

CARRIER CLASS DATA CENTERS

Highly Secure Guarded Facility
24/7/365 Network Operations Center
24/7/365 Technical Support
Redundant Conditioned Air Systems
Redundant Fiber Entry Points
Multiple Uninterruptible UPS Systems
Multiple 1250 KW Generators Onsite
VESDA Early Warning Smoke Detection

Robert Marsh, Head Surfer

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from $299*

Instant Activation!

Dedicated Server

Dual Xeon 2.4 GHz
2 GB RAM • 2 x 73 GB SCSI HD
Remote Console • Remote Reboot
2000 GB Monthly Transfer Included

Over 20,000 Servers!

1-800-504-SURF | ev1servers.net

PLESK7 RELOADED
Preferred Control Panel

IP Compliant. Price subject to change. Quantities Limited.
*Per month. Set-Up fees apply. See web site for complete details.

JavaCaller: The Last Session Bean

A powerful and universal EJB

by Michael Havey

Most Enterprise JavaBeans (EJBs) serve a definite purpose, performing a specific set of actions on behalf of client applications. The ubiquitous Bank Account bean, which supports basic account transactions such as withdrawal and deposit, appears in almost every J2EE tutorial. Students are persuaded that real-life EJBs, though more advanced, are as practical and particular as those developed in the classroom.

This article presents an unusual stateful session bean called JavaCaller, whose interface is entirely general, providing client applications with the ability to run arbitrary Java code on the application server. Because of its generality, JavaCaller can be made to do anything that any other session bean can do; to use the language of computability theory, any problem that can be solved by a session bean can be solved by JavaCaller.

What distinguishes JavaCaller from other session beans is that its execution of actions is completely controlled by client applications. Different clients can make JavaCaller do different things because the client manages the control flow. Supporting a simple reflection interface, JavaCaller exposes methods that let clients create objects and classes, invoke class or object methods, and get or set object fields, all on the server side. As Figure 1 shows, JavaCaller is like a robotic arm, controlled remotely by client applications to manipulate classes and objects in the application server's JVM.

Not that the tutorials are wrong; the specificity of the typical EJB is a virtue. Lack of specificity implies lack of focus and gets flagged during design review; an EJB should do exactly what it is required to do. However, JavaCaller's purpose is atypical: it is a hook into the server JVM, or, alternatively, a shell that runs scripts and jobs on the server. It opens up the application server to users to the degree that operating system shells open up the operating system. True, application servers provide management consoles and open management APIs such as SNMP and JMX, but only for the administration of managed objects, such as applications and resources. The shell is an interface to all the nonmanaged objects too.

JavaCaller API

JavaCaller is a stateful session EJB that can run on any J2EE application server supporting EJB 2.0. (It has been tested on BEA WebLogic 7.0 and 8.1.) JavaCaller's home interface is a create() method with no arguments. Its remote interface, shown in Figure 2, provides client applications with the ability to manipulate objects and classes in the server's address space.

JavaCaller can get or set static fields (getClassField(), setClassField()) and call static methods (callClassMethod()), as well as get and set object-level fields (setField(), getField()) and call object-level methods (callMethod()). JavaCaller can also instantiate a class (instantiate()), add a new class to JavaCaller's class loader (createClass()), and return to the client a serialized copy of an object (getObject()).

JavaCaller stores object and class references in its object cache. This cache, which constitutes the client state of the bean, is empty when the client's session begins and builds up as the client uses objects and classes. JavaCaller is stateful because it manages an object cache for the client. The population of the cache is described in Table 1.

In keeping with the notion of "remote control," the client never uses server objects directly, but has JavaCaller use them on its behalf. The client normally resides in a separate address space anyway, making direct manipulation impossible. JavaCaller's getObject() method returns to the client a serialized copy of a server object, which provides the client with a snapshot of the object's state and perhaps the ability to play with it locally. However, because the object is only a copy, nothing the client does with it has any server-side effect. If the client wants to perform successive operations on the same server-side object, it calls successive JavaCaller methods passing the same numeric

Method	Action
LoadClass	JavaCaller generates a class reference for the specified class name.
CreateClass	JavaCaller creates a new class as defined by client-specified byte code, and then adds a reference to the new class to the cache.
Instantiate	JavaCaller instantiates an object and adds the object reference to the cache.
CallMethod, CallClassMethod	If the method returns an Object (rather than a primitive type), JavaCaller adds its reference to the cache.
GetField, GetClassField	If the field is an Object (rather than a primitive type), JavaCaller adds its reference to the cache.

Table 1 JavaCaller cache management

Michael Havey is a BEA consultant with eight years of industry experience, mostly with application integration. He is also a father, poet, philosopher, and sports fan.

mhavey@bea.com

object reference, representing an index to JavaCaller's object cache. *Note:* Objects maintained by JavaCaller need not be serializable, but getObject() works only for serializable objects.

Most JavaCaller methods return either an integer or a Union. The integer is a cache reference identifier that the client can use to specify an object or class when invoking subsequent JavaCaller methods. The concept underlying the Union type comes from the "C" union type: it represents a value of one type selected from a set of possible types. In the context of JavaCaller, these types are:

- Primitive types, such as int and double, which Union stores as Java wrapper types such as java.lang.Integer and java.lang.Double.
- Cache object references.
- Objects – a convenience for clients that can be used to pass an object argument, rather than an object reference, when instantiating an object, calling a method of an object or class, or setting a field of an object or a class. This option should be used only for serializable objects, such as java.lang.String.

A Union is returned from the getField(), getClassField(), callMethod(), and callClassMethod() JavaCaller methods. If the given field or method return value is an object, JavaCaller sets the object reference field in the Union; otherwise, it sets the primitive field in the Union.

The Union type is also used as an argument type in JavaCaller methods. In setField() and setClassField(), the client uses it to specify a new object or primitive value for the specified field. In callMethod(), callClassMethod(), and instantiate(), the client uses it to specify object or primitive argument values for the given method or constructor. In this case, the client also specifies the string name of the argument type (for example, "int" for a primitive integer type, "java.util.Date" for the Date object type). This is required because classes or objects can overload methods and constructors. Argument types are required to distinguish methods or constructors with the same name.

Intended Code	Code Required
ExecuteThreadManager mgr = Kernel.getExecuteThreadManager();	Union uTM = caller.callClassMethod("weblogic.kernel.Kernel", "getExecuteThreadManager", null);
int numIdle = mgr.getIdleThreadCount();	Union uNumIdle = caller.callMethod(uTM.getObjectRef(), "getIdleThreadCount", null);
StringBuffer sb = new StringBuffer("Num idle = ");	Union sbArgs[] = new Union[1]; sbArgs[0] = new Union(); sbArgs[0].setObject("Num idle = "); sbArgs[0].setArgType(String.class.getName()); Int sbref=caller.instantiate(StringBuffer.class.getName(), sbArgs);
sb.append(numIdle);	Union apArgs[] = new Union[1]; apArgs[0] = new Union(); apArgs[0].setPrimitiveWrapper(uNumIdle.getPrimitiveWrapper()); apArgs[0].setArgType("int"); caller.callMethod(sbref, "append", apArgs);
String s = sb.toString();	Union tos = caller.callMethod(sbref, "toString", null);
System.out.println(s);	Union uout=caller.getClassField(System.class.getName(), "out"); Union printArgs[] = new Union[1]; printArgs[0] = new Union(); printArgs[0].setObjectRef(tos.getObjectRef()); printArgs[0].setArgType(String.class.getName()); Caller.callMethod(uout.getObjectRef(), "println", printArgs);

Table 2 IdlerWatchdog code vs intended server code

The createClass() method allows the client to add a new class on the server side. The client passes a buffer of bytes, representing the contents of a Java class file on the client side. JavaCaller dynamically adds a new class on the server side based on the specified byte code and adds a reference to the class in its object class. The client can then manipulate the new class, or create instances of it and manipulate those instances.

Examples

Example 1: WebLogic Monitoring

This simple example shows how a client application uses WebLogic server objects to monitor the idle thread count of a WebLogic server. As shown in Listing 1, this example gets the idle thread count of the server using the weblogic.kernel package. It then prints out this result in the standard output of the server. (Listings 1 and 2 can be downloaded from www.sys-con.com/java/cfm.)

The main steps are as follows:

1. In the constructor (lines 18–36), get the initial context and acquire a reference to the JavaCaller home interface.

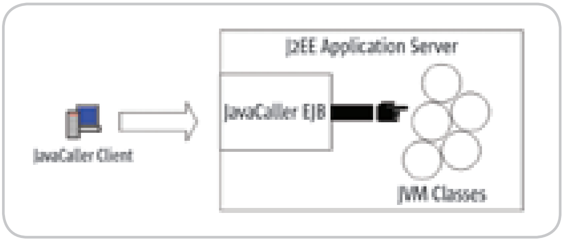


Figure 1 JavaCaller, server JVM robotic arm with remote control

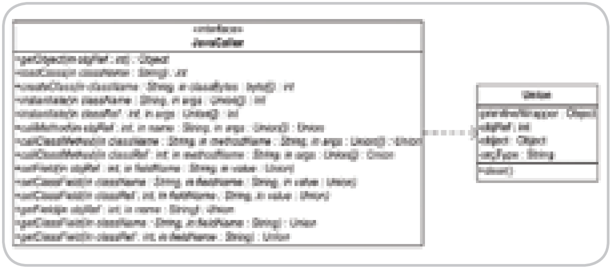


Figure 2 JavaCaller class diagram

2. Call the create() method of the home interface (line 42) to create a new instance of JavaCaller called "caller."
3. In lines 43–95, use caller to affect object interactions on the server side. Table 2 maps server code to equivalent client-side code.
4. Remove the caller, cleaning up the cache (line 99).

Example 2: The Last Web Service

The second example shows how to write a Web service, called PourJavaOnServer, that loads and runs classes on the server. The Web service is written in WebLogic Workshop, a graphical Web service development tool from BEA.

PourJavaOnServer has one method, runObject(), that does the following:

1. Loads a class into the server JVM, based on the fully qualified class name and byte code (passed to the Web service in Base 64 encoding) specified by the user. The class is required to have a default constructor and a method called exec() whose signature is:

```
String exec(String)
```

2. Instantiates the class with its default constructor.
3. Calls the constructed object's exec() method, passing an argument specified by the Web service user.
4. Returns the exec() return string to the Web service user.

The runObject() method, whose source code is provided in Listing 2, calls the JavaCaller EJB to do the hard work, invoking createClass() to load the class into the server JVM (line 4), instantiate() to invoke its default constructor (line 6), callMethod() to invoke the exec() method (line 13), and getObject() (in line 15) to get a local copy of the String object that is returned by the exec() method; that String becomes the return value of the Web service method.

PourJavaOnServer is simply a Web service façade for JavaCaller. It can do what any other Web service deployed on WebLogic Server can do, and anything else too. The user drives the behavior of the Web service; the logic is in the user's Java class. To run a job on the server, the user first writes the source code of the class (it must have a default constructor and exec() method as specified above), then compiles the class. The user then calls the Web service, which uploads the class to the server and executes it. The universality of the JavaCaller EJB implies the universality of the PourJavaOnServer Web service.

Figure 3 shows PourJavaOnServer open in WebLogic Workshop.

Security

Continuing with the analogy of JavaCaller and an operating system shell, when a user logs into (or authenticates with) an operating system, the user's access (or authorization) is dictated by permissions on files. Operating systems define super users who have full access to files, and who therefore can do anything; most users have limited capabilities, however. To authenticate with JavaCaller means to authenticate with WebLogic and acquire a reference to JavaCaller. However, once a user is into JavaCaller, it is impossible to restrict access to most of the server Java objects. J2EE security cannot lock down arbitrary server objects the way operating systems can lock down files. A hacker could do a lot of damage.

The following guidelines should be considered when configuring an application server running JavaCaller.

1. Set up J2EE security on JavaCaller so that its use is restricted to J2EE system administrators. That way, a client application can have access by first authenticating as an administrative user.
2. Restrict access to JavaCaller to users on the internal network.
3. Run the application server hosting JavaCaller under an operating system account that has limited system privileges, preferably just enough for the server to function. JavaCaller can be programmed, for example, to use the "java.io" package to recursively delete all files in a file system; but if the application server lacks permission to delete most of these files, the Java calls made by JavaCaller fortunately will fail.

Summary

The generality and passivity of JavaCaller makes it extremely powerful. However JavaCaller is intended mainly for power users (e.g., administrators, developers, hackers) who are experienced in the terrain of Java and the application server. JavaCaller is an exquisite "back door" that lets experts query or program the server without having to add specialized server-side interfaces. In addition, the control flow of JavaCaller cannot be shared among clients; execution is determined on a per-client, per-session basis. A client cannot deploy its logic for use by the rest of the world; that is not the purpose. JavaCaller is not the last session bean.

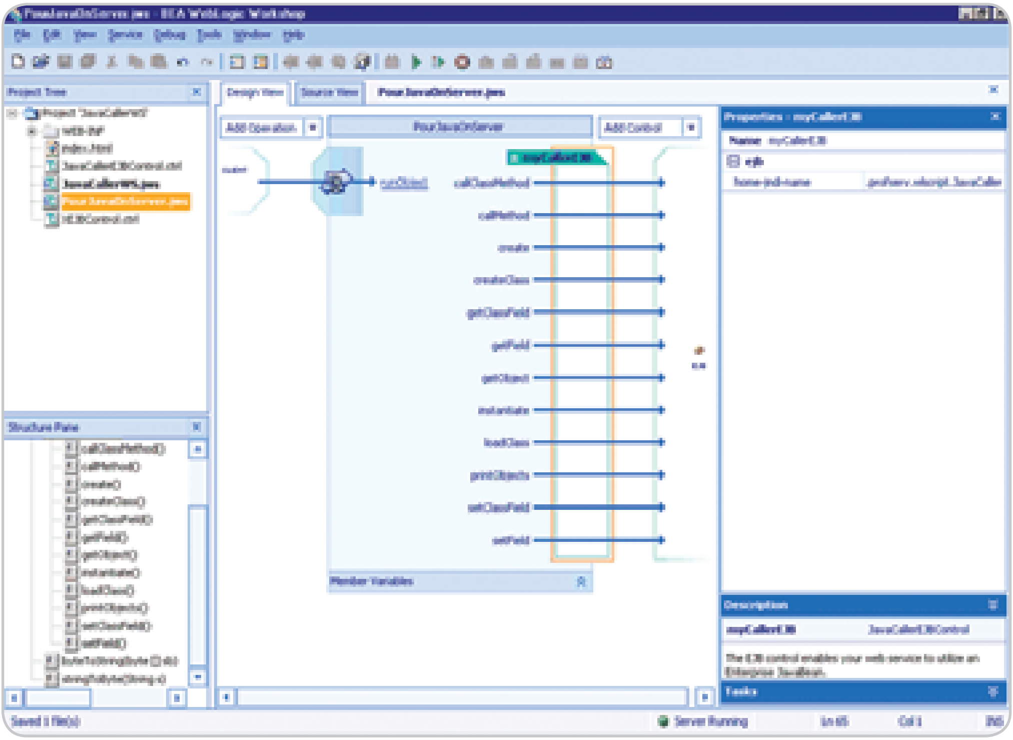


Figure 3 PourJavaOnServer in WebLogic Workshop Editor

21 WAYS TO USE SPREADSHEETS IN YOUR JAVA APPLICATIONS

A free offer for readers of *Java Developer's Journal!*

Formula One e.Spreadsheet Engine: Finally, there's a *supported*, Pure Java tool that merges the power of Excel spreadsheets and Java applications.

1 Automatically generate dynamic Excel reports. No more manual querying and cutting-and-pasting to create Excel reports!
2 Manage calculations and business rules on J2EE servers with Excel files. No more translating Excel formulas to Java code!
3 Embed live, Excel-compatible data grids in applets and Java desktop applications. No more static HTML or presentation-only data grids!

Download this quick-read white paper and trial today!

Download your trial and test our demos and sample code. See for yourself how the Formula One e.Spreadsheet Engine can help your Java application leverage the skills of Excel users in your business.

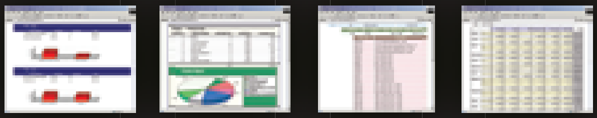
http://www.reportingengines.com/download/21ways.jsp

FREE TRIALS, DEMOS, & SAMPLE CODE!

ReportingEngines™
JAVA REPORTING TOOLS FROM ACTUATE

888-884-8665 • www.reportingengines.com
sales@reportingengines.com

Need to deliver reports from your J2EE application or portal server? Try the Formula One e.Report Engine!



Build reports against JDBC, XML, Java objects, BEA Portal Server logs, BEA Liquid Data, and other sources visually or with Java code. It's embedded! No external report server to set up. Unlimited users and CPUs per license.

http://www.reportingengines.com/download/f1ere.jsp

Portal Standards

The answer to portal interoperability?

by Sue Vickers

As demonstrated by the emergence of multiple portal initiatives within organizations today, the benefits of enterprise portals are clearly understood. It's common to see several enterprise portal platforms deployed throughout an organization. However, many companies are attempting to standardize on one portal framework but are challenged with integrating disparate portal instances. Each portal instance requires developers, partners, system integrators, and independent software vendors (ISVs) to develop portlets using proprietary application program interfaces (APIs) that support only a single portal platform (see Figure 1). Thus, portlets built for one portal platform will not interoperate with another portal platform.

Developers find themselves building the same portlet many times to support the APIs of multiple portal vendors. More important, developers are not the only ones affected by proprietary APIs. Portal page designers and administrators looking to build an enterprise portal are often faced with a limited number of available portlets from a particular portal vendor. The solution to these challenges lies in the form of two complementary portlet standards: Web Services for Remote Portlets (WSRP) and Java Specification Request (JSR) 168, which enable the development of interoperable portlets. With these standards, portlets built for one portal platform can be rendered on a different portal platform as long as the vendor supports one or both of the standards.

The availability of these standards dynamically changes the landscape of the portal market. When deploying enterprise portals, organizations will have a wide array of standards-based portlets to choose from. Portlet providers won't need to invest resources and build APIs for each vendor platform (see Figure 2). In short, building portal pages becomes as simple as selecting portlets from a portal repository or catalog.

Sue Vickers is a group manager for Oracle Application Server Portal and has been with Oracle since 1998. Her team specializes in declarative and programmatic portlet development as well as application integration. She works closely with the Oracle development team that actively participates on the committees for JSR 168 and WSRP. A certified DBA, Sue has considerable experience at Oracle. She joined the Portal team from Oracle Support, where she was the WebDB team lead and an award-winning analyst and gained a breadth of knowledge in Internet and mobile-application development.

sue.vickers@oracle.com

Overview of WSRP and JSR 168

WSRP is a Web services standard that allows the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Because it's a standard, WSRP enables interoperability between a standards-enabled container and any WSRP portal. JSR 168 is a specification that defines a set of APIs to enable interoperability between portlets and portals, addressing the areas of aggregation, personalization, presentation, and security.

Similar to other industry standards, WSRP and JSR 168 are closely related. What is the relationship between WSRP and JSR 168? WSRP is governed by the OASIS technical committees and JSR 168 is governed by the Java Community Process (JCP). These two standards work in conjunction to support portlet and portal interoperability. WSRP is a universal communication protocol between portals (consumers) and portlet containers (producers) of any type. JSR 168 is a Java API that standardizes Java portlets and allows developers to build interoperable portlets. It's important to note that one standard does not require the other. A portal can support JSR 168 and not support WSRP. A WSRP-enabled portal can consume portlets of any type including Java and .NET. The producer simply needs to provide a set of Web services interfaces that include self-description, markup, registration, and portlet management. Developers who build producers based on .NET can register their portlet on the same portal as a developer who has registered a Java producer, as long as that portal supports WSRP.

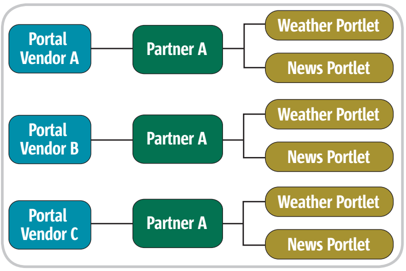


Figure 1 Partner A re-creates the Weather and News portlet for each portal vendor they support.

For example, two developers, developer A and developer B, have built portlets based on at least one of the standards. In addition, there are two portals: portal1 uses WSRP to communicate to portlets remotely (supports WSRP and JSR 168) and portal2 supports only local portlets and cannot communicate with portlets remotely (supports JSR 168, but not WSRP).

Developer A creates a portlet based on JSR 168 and hosts this portlet on an application server. The developer wants to render this portlet on two portals. Developer A provides a WSDL URL to portal1 to register the portlet and add it to a page. Portal1 doesn't care that this portlet is Java since the portlet is remote and communication is done through WSRP. Therefore, its implementation language doesn't matter. Developer A then deploys the portlet into portal2 locally and maps directly to the portlet, rendering the content by making direct calls to it. This developer was able to create the portlet once and make it available to two different portals, although one supported WSRP and JSR 168, while the other supported only JSR 168.

On the other hand, developer B creates a portlet based on .NET, but has deployed it to a container that is WSRP enabled. The developer is able to render this portlet on portal1 since the container is WSRP enabled and the portal does not care that the portlet is implemented in .NET. The developer cannot render the portlet on portal2 because this portal does not support WSRP or .NET.

WSRP provides:

- A WSDL interface description for the invocation of WSRP services

- Markup fragment rules for markup emitted by the WSRP services
- The method to publish, find, and bind WSRP services and metadata
- Interactive, user-facing Web services to be easily plugged into standards-compliant portals
- Parameters that developers and page designers use to create and publish their content and applications as user-facing Web services
- Ways in which page designers and portal administrators find Web services through a catalog, repository, or UDDI and publish them directly into their portal without any programming
- Portlets to make content available for consumption by other portals as Web services
- Standards that allow portlets of different languages to be rendered equivalently on any portal server that supports WSRP
- Support for remote portlets or the ability to store the application in one central location and render at multiple portal servers

JSR 168 provides:

- The portlet API (portlet container) that provides a runtime environment to invoke portlets. Developers use this API to program portlets that render on compliant portals.
- A URL-rewriting mechanism for creating user interaction within a portlet container.
- Security and personalization of portlets.
- A common API for interoperability and portability of Java portlets between portal vendors.

Portal Standards Free Up End Users

With the approval of WSRP in September 2003 and JSR 168 in October 2003, an overwhelming power was passed to end users. They now have the freedom to build portal applications without being tied to a portal vendor's platform. These standards force portal vendors to focus more on application-level features and functionality. Portal vendors must now compete on the basis of their total offering, including high availability, performance, and integration with other components that offer such features as caching, security, and wireless support.

Comparison Between Proprietary APIs and Standard APIs

The portal standards provide a full set of features and functionality, but may not offer all of the capabilities that are available from other vendors' proprietary APIs. The strength that these standard APIs have over proprietary APIs is compatibility. In addition, portal standards also offer the following capabilities:

- Edit defaults
- Customization
- Expiry-based caching
- Private parameters and events
- URL rewriting
- Single sign on (SSO) integration
- Resources "proxying"
- Internationalization

What should consumers consider when deciding which APIs to begin building upon today? Here are some considerations:

- Investigate functionality available versus "must-have" requirements. Can I live with the functionality the standards currently have?
- Is supporting multiple portal platforms really a requirement? This requirement is especially important for portal partners, ISVs, and system integrators, but not always important to other portal consumers. If not a heterogeneous environment, how often should you change portal platforms?
- Remember that vendor APIs are often backward compatible. Using your vendor's APIs today doesn't prevent you from upgrading tomorrow.

The standard committees are already looking at the next release of these standards: WSRP v2.0 is planned for early to mid 2005.

Building Standards-Based Portlets

Many vendors today support JSR 168 and WSRP in many different forms. Developers have multiple options for building standards-based portlets, deploying to a suitable container, and testing on a portal server that supports the standards. As an example, let's look at building JSR 168 portlets and rendering them on portals that support WSRP.

The example below uses a portlet wizard that runs on top of a portlet container. The wizard simplifies portlet development by creating a "hello" portlet skeleton and provides the basic import statements and methods for rendering the application as a portlet. The result is the same as writing a "hello world" portlet manually and is very similar to any standard JSP, with a few differences such as an import of javax.portlet, use of PortletConfig, and portlet styles (portlet-font). The end result of this portlet is "Welcome, this is the show mode" (see Listing 1).

Once this JSP is deployed to a JSR 168 portlet container, it is interoperable. The nice feature about WSRP is the remote portlet capability. The portlet is deployed in one central location and available to multiple portals at the same time without moving the application to each portal server. The portal server communicates to the producer using a WSDL URL (see Figure 3). An example is http://myserver.mydomain.com:8888/myapp/portlets?WSDL.

Using the WSDL URL, with an output similar to the one below, the portal server can ascertain registration properties and configuration mappings (see Listing 2).

Now it's easy for developers to add their business logic to the portlet and build a fully functional enterprise application. The Java code is the most important piece from a development perspective; however, from a deployment perspective, it's important to note the definition file that defines and describes the portlets. This file is called portlet.xml and Listing 3 shows how this file maps the container to your applications.

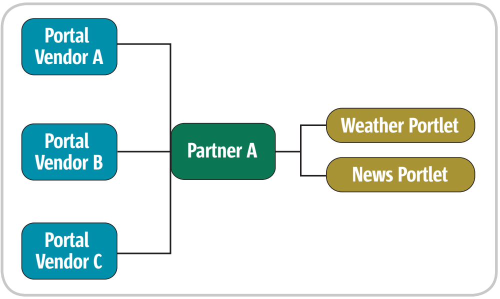
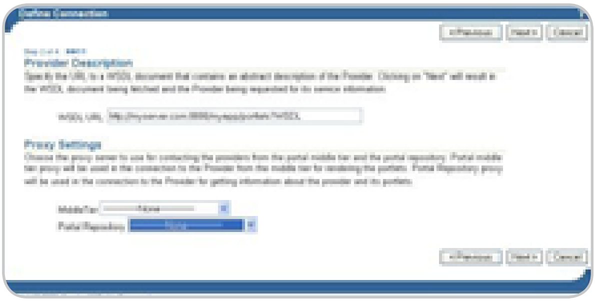


Figure 2 Partner A creates the Weather and News Portlet once and it's accessible to all of the portal vendors they support.



[illegible]

Figure 3 Registration of remote producers using WSRP registration page

Considerations for Enterprise Portal Implementations

Portal vendors will need some time to incorporate standards specifications into their particular solutions. Thus, what should organizations consider when evaluating a portal solution? How should they prepare to migrate to a standards-based portal?

First, organizations must consider portal vendors that actively participate in and support both standards. Most likely, the participating vendor's requirements will be embodied within the specifications. Second, companies must consider a portal solution whose architecture supports federated portals and resembles that of a standards-based portal. Finally, organizations must look for vendors that support both standards-based and nonstandards-based portlets within their portal offerings. Companies will want to preserve their existing investment in portlets and not be burdened with huge development costs when migrating over to a standards-based enterprise portal.

Conclusion

The proliferation of multiple portal platforms within an organization has made it difficult to integrate and standardize into one portal framework. Portal standards such as WSRP and JSR 168 provide organizations with a solution for enabling portal interoperability. These complementary standards provide the facilities that allow portlets built for one platform to be reused within another standards-based portal deployment.

Companies that are evaluating portal products must consider vendors that offer support for these standards and thus avoid being locked into proprietary systems. By deploying standards-based enterprise portals, organizations can not only leverage existing investments and reduce development costs, but also extend their portal functionality using other standards-based technologies.

Resources

- *OASIS/WSRP committee and specifications:* www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp
- *JCP public JSR 168 site:* www.jcp.org/en/jsr/detail?id=168

Listing 1

```
<%@ page contentType="text/html"
import="javax.portlet.*,java.util.*,hellopackage.HelloPorltet,hello
package.resource.Portlet1Bundle"%>

<%
  PortletConfig portletConfig = (PortletConfig)
       request.getAttribute("javax.portlet.config");
  RenderRequest renderRequest = (RenderRequest)
       request.getAttribute("javax.portlet.request");
  RenderResponse renderResponse = (RenderResponse)
       request.getAttribute("javax.portlet.response");
%>

<p class="portlet-font">Welcome, this is the <%= renderRequest.
getPortletMode().toString() %> mode.</p>
```

Listing 2

```
<?xml version="1.0" ?>
<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:
soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:bind="urn:oasis:
names:tc:wsrp:v1:bind" xmlns:wsdl="http://schemas.xmlsoap.org/
wsdl/" targetNamespace="urn:oasis:names:tc:wsrp:v1:wsdl">
<import namespace="urn:oasis:names:tc:wsrp:v1:bind"
location="http://www.oasis-open.org/committees/wsrp/specifications/
version1/wsrp_v1_bindings.wsdl" />
<wsdl:service name="WSRPService">
<wsdl:port binding="bind:WSRP_v1_Markup_Binding_SOAP"
name="WSRPBaseService">
<soap:address location="http://portalstandards.domain.com:80/wsrp/
jaxrpc/WSRPBaseService" />
</wsdl:port>
<wsdl:port binding="bind:WSRP_v1_ServiceDescription_Binding_SOAP"
name="WSRPServiceDescriptionService">
<soap:address location="http://portalstandards.domain.com:80/wsrp/
jaxrpc/WSRPServiceDescriptionService" />
</wsdl:port>
<wsdl:port binding="bind:WSRP_v1_Registration_Binding_SOAP" name="W
SRPRegistrationService">
<soap:address location="http://portalstandards.domain.com:80/wsrp/
jaxrpc/WSRPRegistrationService" />
</wsdl:port>
<wsdl:port binding="bind:WSRP_v1_PortletManagement_Binding_SOAP"
name="WSRPPortletManagementService">
<soap:address location="http://portalstandards.domain.com:80/wsrp/
jaxrpc/WSRPPortletManagementService" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Listing 3

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-
app_1_0.xsd" version="1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/port-
let-app_1_0.xsd http://java.sun.com/xml/ns/portlet/portlet-app_1_
0.xsd">
  <portlet>
    <description xml:lang="en">A very simple portlet.</descrip-
tion>
    <portlet-name>HelloWorld</portlet-name>
    <display-name xml:lang="en">Hello World Portlet</display-name>
    <portlet-class>oracle.portal.portlet.sample.HelloWorldPortlet</
portlet-class>
    <expiration-cache>-1</expiration-cache>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>edit</portlet-mode>
      <portlet-mode>help</portlet-mode>
      <portlet-mode>about</portlet-mode>
      <portlet-mode>preview</portlet-mode>
      <portlet-mode>edit_defaults</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>Hello World Portlet</title>
      <short-title>Hello</short-title>
      <keywords>Hello, World, Portlet</keywords>
    </portlet-info>
  </portlet>
. . . . .
</portlet-app>
```

NEW VERSION

InstallShield®

✓ Linux

✓ Unix

✓ Mac OS X

✓ Windows

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

Download Now!

www.installshield.com/JDJ

Special Offer!
Receive expert training with purchase.
See Web site for details.

macrovision®

InstallShield, a Macrovision Company

SOA Solutions with J2EE

Using different technical and functional requirements

by Bruno Collet

Throughout this article I'll describe how an effective service-oriented architecture (SOA) can be achieved using J2EE technologies. In particular, I'll focus on which J2EE component types and communication channels to choose according to specific, real-world situations.

Description of the Example System

To illustrate an SOA system made of J2EE technologies, nothing is better than a good old example. Figure 1 depicts the main situations that can arise in a realistic SOA system. A component stereotype indicates the type of client or service; a dependency stereotype indicates the communication type, either as a technology (such as Web services) or as a protocol (such as CORBA IIOP); and a no dependency stereotype indicates RMI communication.

The system allows client systems to process an order (service OrderWorkflow), manage customers (service Customer), and produce reports (service Reporting). The OrderWorkflow service checks product availability and retrieves product details (service Product), creates a new customer if needed (service Customer), and orders the selected products (service Order). The Order service persists the order (service OrderData), manages the payment (service Payment), and triggers some process in a legacy system (service BackOfficeOrder).

Service Layers

Services differ according to multiple criteria. Whether the service is business oriented, public or private, or stateful or stateless dictates what layer the service is in and ultimately how it is implemented and what interfaces it exposes.

Top-layer services such as OrderWorkflow are coarse-grained business services, often stateful, that depend on one or more finer-grained stateless business services. Bottom-layer services are fine grained and data oriented, not business oriented.

Services are also separated into public and private services. Public services are services that are available outside of the system, and possibly outside of the organization. They are typically services that have a business meaning. Private services have no business meaning; they exist to support business services and there's no point in making them available to other systems. In Figure 1, public services are colored in blue and private services are not colored.

Although the number of layers of an SOA system is arbitrary, we can categorize them and associate usual J2EE component types to them as shown in Table 1.

Service Communication Interfaces

In SOA systems, service interfaces are even more important than service implementations, because interoperability essentially depends on the communication technologies supported by the service interfaces. Remember one of the founding principles of service architecture: the service implementation is separated from the service interface(s).

To continue this discussion, we'll distinguish between "internal" and "external" clients. A client is internal if the organization controls the network path between the client and the service. In all other cases the client is external. This distinction is driven by the fact that firewalls might be located between the external client and the service, and no

Service Layer	J2EE Component Type
Example Business workflow	stateful session EJB
OrderWorkflow Business	stateless session EJB
Order Data and enterprise resources	entity EJB/Data Access Object/resource/adapter OrderData

Table 1 J2EE component type according to service layer

SOA Reminder

SOA promotes loosely coupled, distributable, reusable services that can be invoked through platform-independent service interfaces. A service is a function that is well defined, self-contained, and does not depend on the state of other services.

SOA is not a technology, an implementation, or a design pattern; it is an architectural paradigm. For example, Web services is a technology-enabling SOA implementation. But Web services is not the only way to implement an SOA system and does not guarantee that the system will be SOA-compliant.

Resources

- *SOA Blueprints Concepts:* www.middlewareresearch.com/endeavors/artifacts/soa-blueprints/SOA_Blueprints_Concepts_v0_5.pdf
- *Web Services and SOA:* www.service-architecture.com/index.html

Bruno Collet is a seasoned J2EE architect with five years of experience. He recently founded Studio 184 (www.studio184.com), where he is developing the ApolloNews news aggregator. Bruno holds a masters in computer science from ULB (Belgium), as well as several industry certifications (www.brunocollet.com).

brunocollet@brunocollet.com

assumption can be made on which ports the firewall keeps open and which protocols the firewall lets pass, except that it allows text over HTTP. In particular, this is true if the client component is located on the Internet.

The major condition to decide on in a service interface technology is whether the service clients are internal or not. In Figure 1, external clients are red and internal clients are green. ExternalMainClient uses the Internet. As mentioned earlier, this causes severe restrictions as to which ports and protocols are available for communication. For example, communicating in RMI or IIOP is not realistic, since most firewalls will not let these protocols go through. In such a situation, only text-based protocols over HTTP (or HTTPS) should be considered.

Web services, a standardized technology that leverages the convenience of XML as text over HTTP, is the best technology available, provided that the client platform supports Web services. The .NET client from our example understands Web services. If it were not the case, we would have had to downgrade the communication to, for example, plain XML text over HTTP, with custom XML generation and parsing on the client side.

Now let's have a look at internal clients. OrderClient is a J2EE application client using RMI to communicate with OrderWorkflow. Why RMI and not Web services? Isn't Web services the most cool technology that every supporting platform should use when possible? Well, no. Using Web services between "internal" J2EE components has a heavy performance toll. Performance, which is a critical factor for most systems, is much more favorable to RMI than to Web services. It must be noted that Web services still has opportunities for significant optimizations, but current Web services implementations are notoriously slower than RMI.

The golden rule is: if a J2EE service only has internal J2EE or Java clients, stick to RMI, but if you can't make this assumption, consider Web services. This rule can be understood with a bit of logic: the closer the interface technology is to the implementation technology, the less work that has to be done to translate between the two. For example, XML, which Web services messages are made of, is farther from Java than RMI is. The gap between XML types and Java types is wider than the gap between CORBA IIOP types and Java types, which in turn is wider than the gap between RMI types and Java types. To generalize this rule, we can say that the more interoperable a technology is, the slower it tends to be. Although technology implementations can provide exceptions to this rule, the rule remains true in the vast majority of cases.

Each situation requires a decision based on the trade-off between interoperability and performance.

Fortunately, interoperability and performance benefits can be combined thanks to the fact that a service can have more than one interface. OrderWorkflow is a good example. While its implementation of the application logic is devel-

RMI over HTTP

RMI works over JRMP (Java Remote Methods Protocol) and IIOP. The J2EE specification requires HTTP tunneling, meaning that RMI calls go through HTTP instead of JRMP. As firewalls let HTTP pass through, that solves the communication problem. Unfortunately, the degradation of performance is such that it is better to use Web services than HTTP tunneling. Moreover, HTTPS tunneling is not required in the J2EE specification, which is a serious security issue unless we choose to manage security programmatically.

Resource

- *Access EJBs Through Firewalls:* www.ftponline.com/javapro/2002_09/online/j2ee_asankaran_09_03_02/

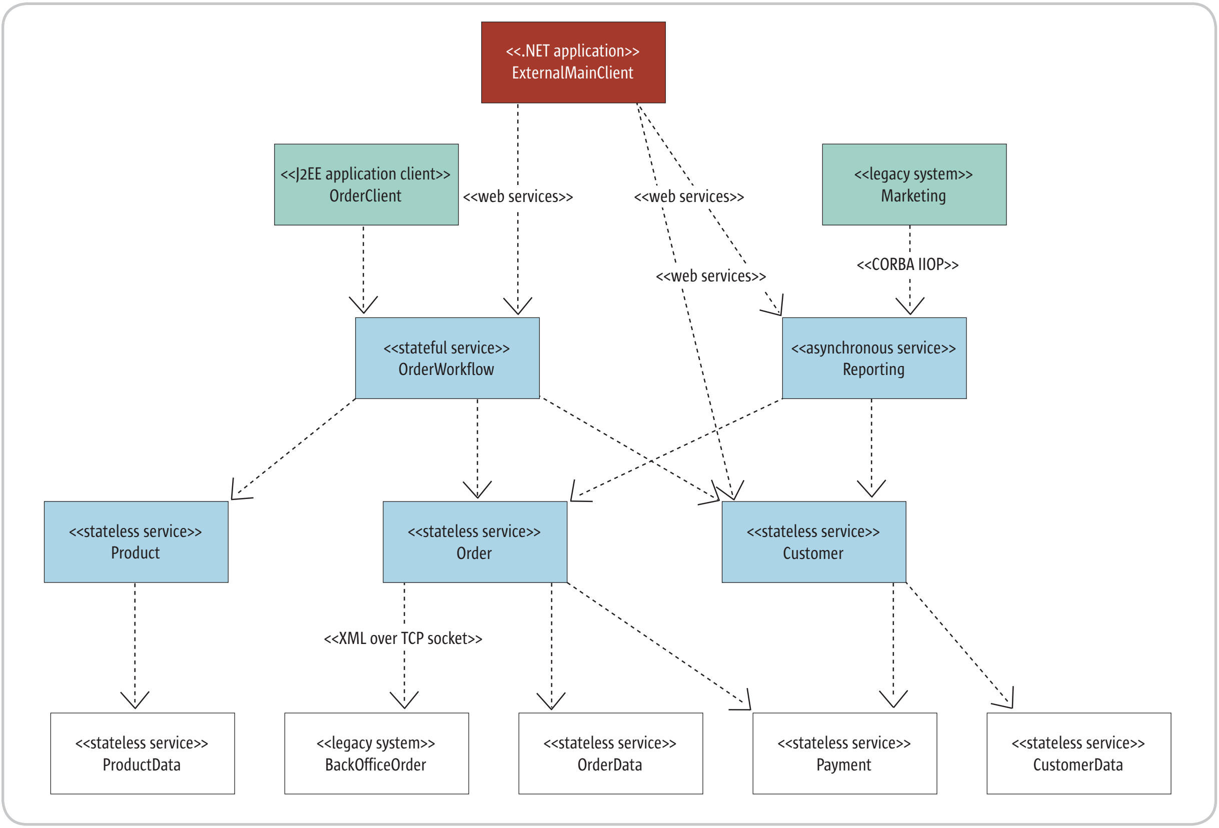

<<.NET application>> ExternalMainClient
<<J2EE application client>> OrderClient
<<web services>>
<<web services>>
<<web services>>
<<legacy system>> Marketing
<<CORBA IIOP>>
<<stateful service>> OrderWorkflow
<<asynchronous service>> Reporting
<<stateless service>> Product
<<stateless service>> Order
<<stateless service>> Customer
<<XML over TCP socket>>
<<stateless service>> ProductData
<<legacy system>> BackOfficeOrder
<<stateless service>> OrderData
<<stateless service>> Payment
<<stateless service>> CustomerData

Figure 1 The system

oped only once, the service presents two different interfaces: one RMI and one Web services.

Moreover, with the right tools, service interfaces can be generated automatically from one another. For example, a tool such as Castor XML or a JAXB implementation can generate Java classes from the XML Schemas. The RMI interface is thus generated with minimal hand coding. Enabling Web services as EJB endpoints is also a good alternative.

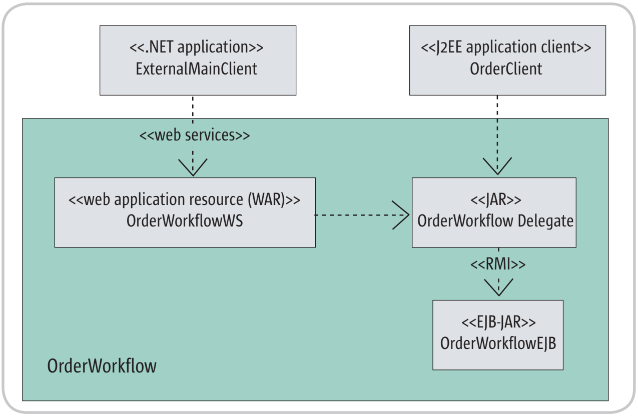

<<.NET application>> ExternalMainClient
<<J2EE application client>> OrderClient
<<web services>>
<<web application resource (WAR)>> OrderWorkflowWS
<<JAR>> OrderWorkflow Delegate
<<RMI>>
<<EJB-JAR>> OrderWorkflowEJB
OrderWorkflow

Figure 2 OrderWorkflow component

CORBA

Common Object Request Broker Architecture might be said to be the first SOA implementation because it embraces the very same goals as SOA. However, it's not as interoperable as Web services is because it doesn't usually go through firewalls and it's not supported by Microsoft platforms (MS developed COM and DCOM as their own flavors of CORBA). CORBA protocol is Internet Inter-ORB Protocol (IIOP). The interfaces to CORBA services are defined in the Interface Description Language (IDL), which maps to popular languages such as C, C++, Java, Ada, and Python. The languages supporting CORBA are interoperable. RMI is based on IIOP but is specific to Java. Therefore it's natural to consider using IIOP before any other protocol if RMI cannot be used.

Resources

- *CORBA FAQ:* www.omg.org/gettingstarted/corbafaq.htm
- *Java, RMI & CORBA: A comparison of two competing technologies:* www.javacoffeebreak.com/articles/rmi_corba/
- *CORBA vs Web Services:* www2002.org/CDROM/alternate/395

Figure 2 focuses on the OrderWorkflow service to illustrate how a single service is accessed through two communication channels. ExternalMainClient accesses the service through its Web services interface, which in terms of J2EE component types is a Web application resource (WAR). Upon receiving a Web services request, the WAR makes an ordinary Java method call to the business delegate located in the JAR, which in turn makes an RMI call to the EJB.

In contrast, the local OrderClient J2EE client directly calls the delegate without going through the WAR. The service business logic is located in the EJB-JAR component, and the WAR implements a Web services interface layer above the ordinary business delegate. Both interfaces share exactly the same business logic. This design pattern ensures a multiple-communication-channel SOA as well as the scalability, distributability, and other capabilities provided by the EJB technology.

The marketing application is a C legacy application that calls the reporting component to present statistical data to the user. Marketing is an internal client, but does not support RMI. CORBA is an effective and mature distributed architecture available to both C and J2EE platforms. Again, the reason for choosing CORBA over Web services is the performance. As a general rule, internal components should not communicate through Web services unless some other capability of Web services, such as the UDDI publish-discover mechanism, for example, is a deciding factor.

Sometimes we have to deal with a component that does not support any high-level communication technology. In this case, only low-level protocols remain. The Order service accesses the BackOfficeOrder legacy system by writing XML text on a TCP socket. BackOfficeOrder is nonetheless a service, but its interface is very restrictive in terms of communication technology and message format. Unfortunately, this situation is rather common. Access to BackOfficeOrder is managed through a resource adapter (Java Connector Architecture technology), a special kind of data source.

Communication Interface Summary

Figure 3 is a chart that summarizes the decision process for selecting an appropriate interface communication technology according to most usual situations. Alternatives such as JINI, HTTP tunneling, middleware systems, and the proprietary technologies of application servers are not covered.

Other Considerations

Asynchronous Services

Until now we've discussed only synchronous services. Indeed,

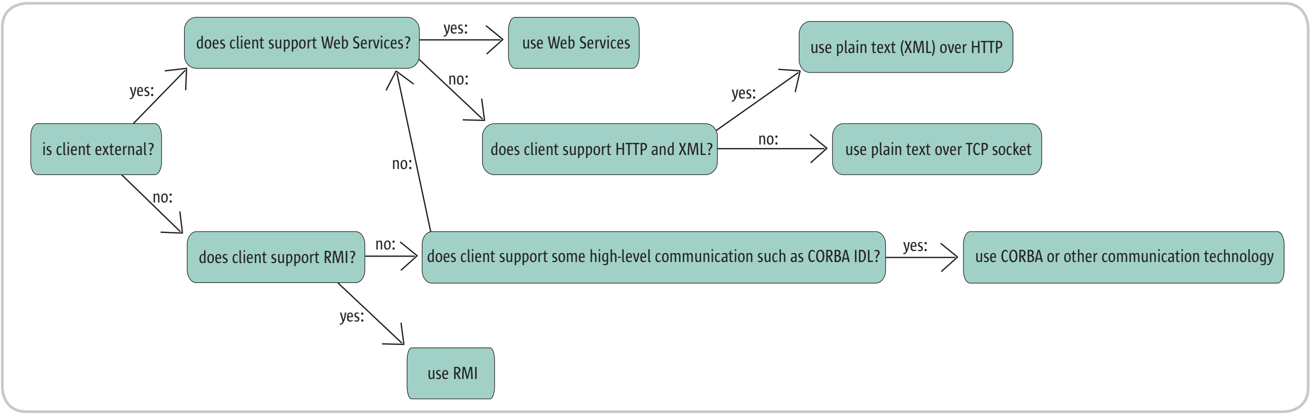
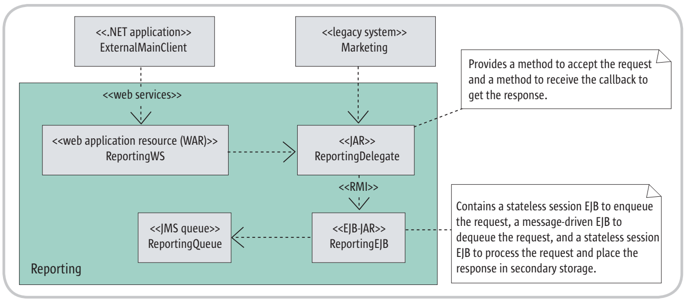

is client external?
yes:
no:
does client support Web Services?
yes:
use Web Services
no:
does client support HTTP and XML?
yes:
use plain text (XML) over HTTP
no:
use plain text over TCP socket
does client support RMI?
yes:
use RMI
no:
does client support some high-level communication such as CORBA IDL?
yes:
use CORBA or other communication technology
no:

Figure 3 Choosing a communication interface



<<.NET application>> ExternalMainClient
<<legacy system>> Marketing
<<web services>>
<<web application resource (WAR)>> ReportingWS
<<JAR>> ReportingDelegate
<<RMI>>
<<JMS queue>> ReportingQueue
<<EJB-JAR>> ReportingEJB
Reporting

Provides a method to accept the request and a method to receive the callback to get the response.

Contains a stateless session EJB to enqueue the request, a message-driven EJB to dequeue the request, and a stateless session EJB to process the request and place the response in secondary storage.

Figure 4 Reporting asynchronous component

RMI, CORBA, and Web services are inherently synchronous. Asynchronous services bring interesting capabilities when the time to perform an operation is likely to exceed the time that the client component is willing to wait. It also allows event-driven processes. Java Message Service (JMS) is the J2EE technology providing asynchronous capability.

In Figure 4, ExternalMainClient calls the Reporting service via Web services, and the marketing legacy application accesses the service via CORBA IIOP. For Reporting, the messaging queue is not visible for the client. Instead, to provide a degree of abstraction, producers and consumers are hidden under Web services and RMI interfaces. To switch from synchronous technologies such as Web services and CORBA to asynchronous JMS, the producer EJB generates a token that is immediately sent back to the client. After the reporting process is completed for a request, the response (the report) is placed in secondary storage (in a database, for example). The client can call back with its token at any time to check whether the report is ready. If it is, the report is sent back to the client. Alternatively, the client component could provide the server with a callback system, so that the server automatically sends the response to the client when the job is done.

Publish and Discover

SOA also promotes publishing and discovery of services. Publishing services is useful when we need very loose client/server relationships. In other words, it's important if the clients don't know where and how to reach the service directly. For example, if we set up a free news service that any client can use, we would publish the service so that any client is able to discover it and get a description of its message formats. In most other situations, the publish-discover mechanism is of little use. In the example system, the client/server relationships are static, as is usually the case in intraorganization systems, and makes publish-discover irrelevant. For Web services, publish-discover is provided by UDDI. No such mechanism exists in RMI, but, although it would not comply with any standard, it's easy to design an entry point that provides clients with the localization and interface descriptions of EJBs and other resources.

Transactions

As a small reminder, atomic transactions are a well-known technique for guaranteeing consistency in the presence of failures. The ACID properties of atomic transactions (Atomicity, Consistency, Isolation, and Durability) ensure that even in complex business applications, consistency of state is preserved despite concurrent accesses and failures.

In an SOA system transaction, demarcation typically defines whether a service creates a new transaction, uses an existing transaction, or does not support transactions. In our system, OrderWorkflow automatically starts a new transaction and all underlying services (Product, Order, Customer) use it. If at the end of the transaction the process of creating the order fails, the creation of the customer is rolled back and data integrity is preserved.

Support for transaction management differs according to communication technologies. RMI does support all that is needed, so that is again a strong point of RMI. Web services does not support transactions out of the box. There are, however, proprietary extensions, and a couple of extensions-in-the-making by Sun, OASIS, and others. This restriction means that a transaction started by the Web services client cannot be propagated to the Web services service, but the service is still able to start its own transaction as OrderWorkflow does.

RMI is CORBA-based; it's theoretically feasible to propagate a CORBA transaction to an RMI service and vice versa, but the lack of support for the specification makes it difficult at best.

To make things worse, it is notorious that transactions management across J2EE servers from different vendors is not seamless, due to the lack of compliance with the J2EE specification.

Security

J2EE provides authentication, authorization, and encryption. We can thus build secure channels between J2EE components.

For the same reason as for transaction management, interoperability of security mechanisms between CORBA and J2EE is far less obvious than the theory suggests.

As for Web services, current security capabilities are those of HTTP, because Web services works over HTTP. It means that Web services security can make use of SSL authentication and encryption, and this is enough for most business applications. When reaching a J2EE container, the container authenticates the user/password received via HTTP and retrieves user credentials to check authorization when the transaction wants to access protected resources.

Conclusion

The most important conclusion is that, because it has a significant performance cost, interoperability is not always desirable. To alleviate this hard truth, remember that a service can implement more than one interface, achieving interoperability as well as decent performance for a little extra work.

Also, it must be noted that the component approach underlying J2EE is well suited for SOA. For RMI services in particular, applying common design patterns such as session facade, business delegate, and service locator to EJB-based services enables compliance with all SOA principles except interoperability. This makes EJB-based services a great solution for SOA in a J2EE environment.

Even as we have seen that Web services technology is far from being a magic bullet for SOA systems, it is nonetheless the most effective technology to expose services when no assumption can be made as to the platform or the location of the client/server components, which is a common situation for top-layer business services. In other words, Web services is the best solution when interoperability is a requirement.

Resources and Further Readings

- *Exposing EJB Components as Business Services: An Architect's View:* www.oracle.com/technology/pub/articles/davydov_ejb.html
- *SOA Blueprints:* www.middlewareresearch.com/soa-blueprints/
- *Comparing Web Service Performance:* www.sosnoski.com/presents/cleansoap/comparing.html
- *Fast Web Services:* http://java.sun.com/developer/technicalArticles/WebServices/fastWS/
- *Web Services Transaction Management Specification:* http://developers.sun.com/techtopics/ webservices/wscaf/wstxm.pdf
- *RMI/IIOP, nice idea but the reality is turning out to be different:* www.theserverside.com/articles/article.tss?l=RMI-IIOP
- *Web Services Security:* http://webservices.xml.com/pub/a/ws/2003/03/04/security.html

Developing Wireless Bluetooth Applications in J2ME

Portable, secure, and highly usable

by Peter V. Mikhalenko

Mobile communication comes into our daily lives very quickly, and as of today several wireless technologies have become standard. In this article I'll briefly review Bluetooth principles and the principles of Java development for Bluetooth on mobile devices.

The Java APIs for the Bluetooth wireless technology (JABWT) standard, defined by the JSR 82 specification (www.jcp.org/en/jsr/detail?id=82), supports the rapid development of Bluetooth applications that are portable, secure, and highly usable. Wireless device manufacturers have responded to the JABWT specification by announcing mobile phones and other products that will run JABWT applications.

What Is Bluetooth?

According to the latest Bluetooth 1.2 specification (www.bluetooth.org/spec/), Bluetooth wireless technology is a short-range communications system that's intended to replace the cable(s) connecting portable and/or fixed electronic devices. The Bluetooth core system consists of a radio frequency (RF) transceiver, baseband, and protocol stack. The system offers services that enable the connection of devices and the exchange of a variety of classes of data between these devices. Actually it's a wireless communication protocol that, like HTTP or FTP, operates in a client/server architecture. It uses the 2.4 GHz band known in the U.S. as the Industrial-Scientific-Medical (ISM) band. The 802.11b wireless LAN protocol operates in this band as well, although they are intended for different needs and don't interfere with each other.

Peter Mikhalenko holds a masters in computer science from Moscow State University. He has made code contributions to several worldwide open source projects and has written many articles on XML and Java. Peter is a Sun certified professional and works in Deutsche Bank co-developing a finance transactional system.

peter@mikhalenko.ru

Bluetooth Architecture

Bluetooth is a completely different way to communicate compared to cables, since during a typical operation a physical radio channel is shared by a group of devices that are synchronized to a common clock. One device provides the synchronization reference and is known as the master. All other devices are known as slaves. A group of devices synchronized in this way form a so-called piconet. A master and a single slave use point-to-point communication; if there are multiple slaves, point-to-multipoint communication is used. A master unit is the device that initiates the communication. A device in one piconet can communicate to another device in another piconet, forming a scatternet. The physical channel is subdivided into time units known as slots. Data is transmitted between Bluetooth devices in packets that are positioned in these slots. Above the physical channel there is a layering of links and channels and associated control protocols. The hierarchy of channels and links from the physical channel upward consists of a physical channel, a physical link, logical transport, a logical link, and a L2CAP channel. In Figure 1 we can see a high-level view of the architecture of the Bluetooth protocol stack.

The radio layer is the physical wireless connection. To avoid interference with other devices that communicate in the ISM band, the modulation is based on fast frequency hopping. Bluetooth divides the 2.4 GHz frequency band into 79 channels, 1 MHz apart (from 2.402 to 2.480 GHz), and uses this spread spectrum to hop from one channel to another, up to 1,600 times per second.

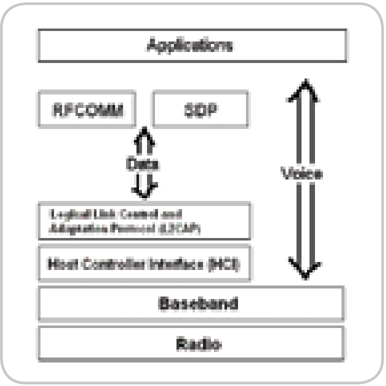

Applications
RFCOMM
SDP
Data
Voice
Logical Link Control and Adaptation Protocol (L2CAP)
Host Controller Interface (HCI)
Baseband
Radio

Figure 1 Bluetooth protocol stack

The baseband layer is responsible for controlling and sending data packets over the radio link. It provides transmission channels for both data and voice. The baseband layer maintains Synchronous Connection-Oriented (SCO) links for voice and Asynchronous Connectionless (ACL) links for data. SCO packets are never retransmitted but ACL packets are to ensure data integrity. The Host Controller Interface (HCI) controls the low-level operation of the radio and is the interface between the Bluetooth device and the host computer. Logical Link Controller Adaptation Protocol (L2CAP) multiplexes all data passing through the Bluetooth device. However, audio signals have direct access to the HCI. The Service Discovery Protocol (SDP) finds services on remote Bluetooth devices. RFCOMM is widely known as the virtual serial port protocol, because it allows a Bluetooth device to simulate the functionality of a serial port.

At the application level it can be any application that needs to exchange data through a wireless connection. Object Exchange Protocol (OBEX) is widely used and allows you to exchange object data (such as files). It's interesting that the Object Exchange (OBEX) API was developed by Java enthusiasts and became a subpackage of JSR 82.

Bluetooth Interacting

Bluetooth profiles exist to ensure interoperability among Bluetooth-enabled devices and applications from different manufacturers and vendors. A profile defines the roles and capabilities for specific types of applications. Bluetooth devices cannot interact unless they conform to a particular profile. There are a number of different profiles in a Bluetooth specification, such as Synchronization, Serial Port, and LAN Access. The Synchronization Profile defines the application requirements for Bluetooth devices that need to synchronize data on two or more devices. The Serial Port Profile defines the requirements for Bluetooth devices that need to set up connections that emulate serial cables and use the RFCOMM protocol. The LAN Access Profile defines how Bluetooth devices can access the services of a LAN using PPP, and shows how PPP mechanisms can be used to form a network consisting of Bluetooth devices.

Security is provided in three ways: pseudo-random frequency hopping, authentication, and encryption. *Frequency hops* make it difficult for anyone to eavesdrop. *Authentication* allows a user to limit connectivity to specified devices. *Encryption* uses secret keys to make data intelligible only to authorized parties.

Java API

Now we can see how easy it is to connect two devices with Bluetooth using Java. The Java Bluetooth API relies on the Java Generic Connection Framework, which limited it to J2ME for a long time. However, now it's been proposed to include GCF in J2SE, and the Bluetooth API can be made accessible for a broader range of systems. The Java APIs for Bluetooth define two packages: javax.bluetooth for the core Bluetooth API and javax.obex for the Object Exchange (OBEX) protocol. Any Bluetooth application has these components: stack initialization, device management, device discovery, service discovery, and communication. According to JSR 82, the underlying Bluetooth system must support a Bluetooth Control Center (BCC), a control panel much like the application that allows a user or OEM to define specific values for certain configuration parameters in a stack. In particular, it will be used in a stack initialization.

Stack Initialization

Prior to starting wireless communication the Bluetooth device should be initialized. This is done in a vendor-dependent way, and exact steps for stack initialization are beyond the scope of the Bluetooth API specification. As shown by Bruce Hopkins, the author of *Bluetooth for Java,* in his article "Getting Started with Java and Bluetooth" (http://today.java.net/pub/a/today/2004/07/27/bluetooth.html), it is done using several settings in the Atinav Java Bluetooth SDK (see Listing 1). It is important that these calls are not part of JSR 82. Other JSR 82 implementations may incorporate other ways to initialize the stack.

Device Management

The JSR 82 API introduces two classes that can be used for device management: LocalDevice to request static information about the Bluetooth device, and RemoteDevice to retrieve information about devices in the Bluetooth neighborhood, e.g., a remote device Bluetooth Address. LocalDevice depends on the javax.bluetooth.DeviceClass class to retrieve the device's type and the kinds of services it offers. The RemoteDevice class represents a remote device (a device within a range of reach) and provides methods to retrieve information about the device, including

Think grid computing is only for specialized applications? Think again!

xTier™/GRID

Grid Computing for Java - Made Simple

Bring the power of GRID to your Java/J2EE applications

Fitech Laboratories Inc.
sales: +1.646.495.5076 or sales@fitechlabs.com
suport: support@fitechlabs.com - 24/7/365

www.fitechlabs.com/grid

its Bluetooth address and name. Every Bluetooth device has a unique hardware address, like the MAC address for computers. You can set the level of device discovery, enabling other Bluetooth devices to find the current device by calling the setDiscoverable() method in the LocalDevice object (see Listing 2).

Device Discovery

Wireless devices need a mechanism that allows them to find other devices and gain access to their capabilities. The core Bluetooth API's DiscoveryAgent class and DiscoveryListener interface provide the necessary discovery services. There are three ways to obtain a list of accessible devices. The DiscoveryAgent.startInquiry() method places the device into an inquiry mode. To take advantage of this mode, the application must specify an event listener that will respond to inquiry-related events. DiscoveryListener.deviceDiscovered() is called each time an inquiry finds a device. When the inquiry is completed or canceled, DiscoveryListener.inquiryCompleted() is invoked. If the device doesn't wish to wait for devices to be discovered, it can use the DiscoveryAgent.retrieveDevices() method to retrieve an existing list. Depending on the parameter passed, this method will return either a list of devices found in a previous inquiry or a list of preknown devices that the local device has told the Bluetooth Control Center it will contact frequently. The most simple approach is shown in Listing 3 where DiscoveryAgent is asked for an object to notify us via the DiscoveryListener interface whenever it detects a new Bluetooth device.

Service Discovery

Service discovery allows you to find nearby services, regardless of what devices are offering them. Because service discovery is much like device discovery, DiscoveryAgent also provides methods to discover services on a Bluetooth server device and to initiate service-discovery transactions. Before a service can be discovered, it must first be registered – advertised on a Bluetooth server device. The server is responsible for creating a service record that describes the service offered, accepting connections from clients, adding a service record to the server's Service Discovery Database (SDDB), etc. In general it works similar to Web services. Listing 4 provides an example of a service registration.

Communication

The two devices must share a common communications protocol. Applications can access a wide variety of Bluetooth services because the Java APIs for Bluetooth provide mechanisms that allow connections to any service that uses RFCOMM, L2CAP, or OBEX as its protocol. If a service uses another protocol (such as TCP/IP) layered above one of these protocols, the application can access the service, but only if it implements the additional protocol in the application using the CLDC Generic Connection Framework. The URL used as a service record consists of digits and symbols and may look something like btspp://508031205080110F1B1B1D1C100:8. This means that a client should use a Bluetooth Serial Port Profile (btspp://) to establish a connection to server channel 8 on a device with the address 508031205080110F1B1B1D1C100. Device addresses are similar to the physical (MAC) addresses of computers. Simple RFCOMM connections look like Listing 5.

Resources

- *JSR 82, Java API for Bluetooth:* www.jcp.org/en/jsr/detail?id=82
- *Bluetooth 1.2 specification:* www.bluetooth.org/spec/
- Hopkins, B., and Antony, R. (2003). *Bluetooth for Java*. Apress.
- *"Getting started with Java and Bluetooth":* http://today.java.net/pub/a/today/2004/07/27/bluetooth.html

Listing 1

```
import javax.bluetooth.*;
import javax.microedition.io.*;
import com.atinav.bcc.*;

  BCC.setPortName("COM1");
  BCC.setBaudRate(57600);
  BCC.setConnectable(true);
  BCC.setDiscoverable(DiscoveryAgent.GIAC);
```

Listing 2

```
// retrieve the local Bluetooth device object
LocalDevice local = LocalDevice.getLocalDevice();
// retrieve the name of the local Bluetooth device
String name = local.getFriendlyName();
```

Listing 3

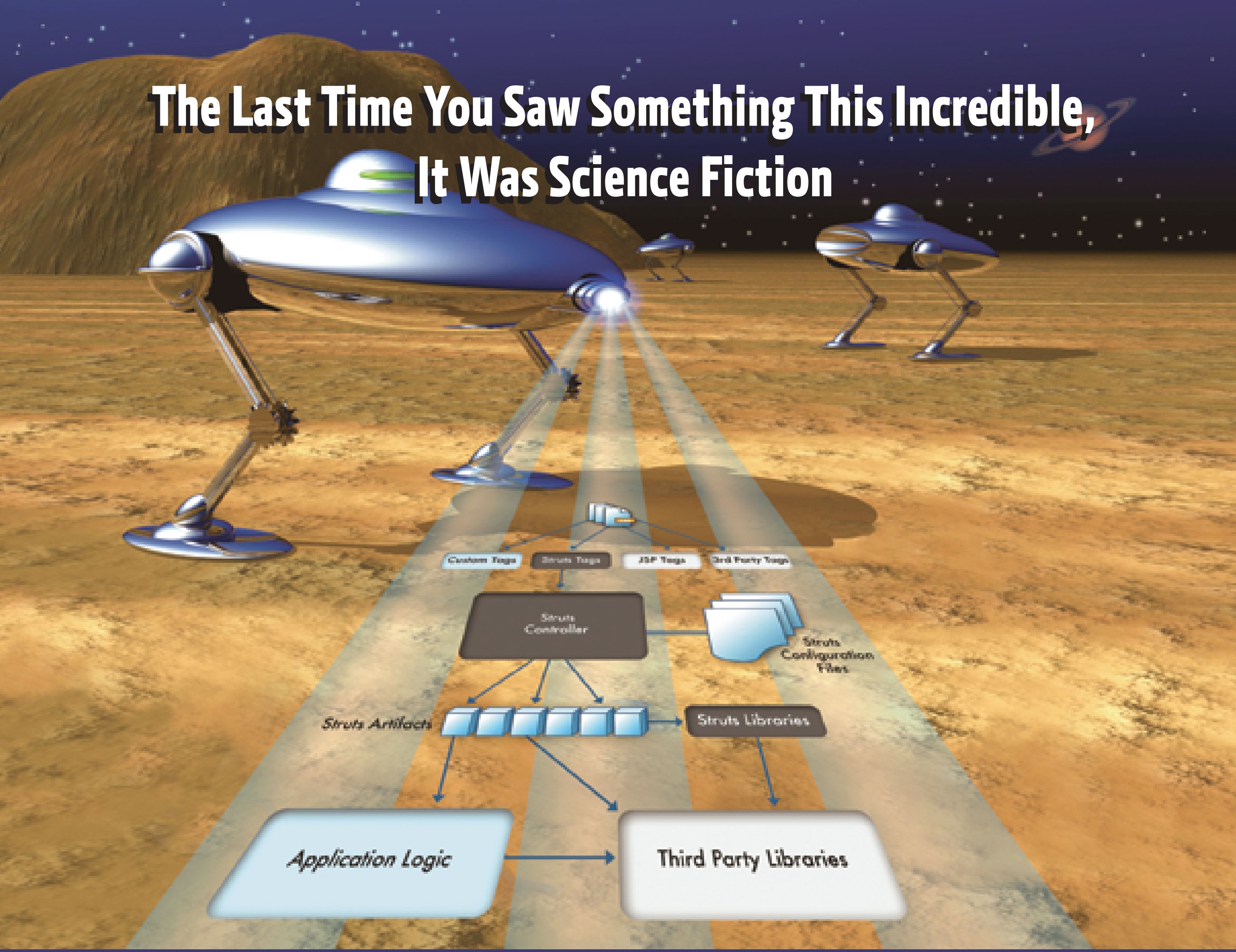
```
LocalDevice localdevice = LocalDevice.getLocalDevice();
DiscoveryAgent discoveryAgent = localdevice.getDiscoveryAgent();
discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
```

Listing 4

```
// Service registration

// invoke Connector.open with a server connection URL argument
StreamConnectionNotifier service =
     (StreamConnectionNotifier) Connector.open("someURL");
// Obtain the service record created by the server device
ServiceRecord sr = local.getRecord(service);

// Indicate that the service is ready to accept a client connec-
tion. acceptAndOpen() blocks
//    until a client connects.
StreamConnection connection =
     (StreamConnection) service.acceptAndOpen();

// DO SOME EXCHANGE HERE

service.close();
```

Listing 5

```
String url =
     serviceRecord.getConnectionURL(
          record.NOAUTHENTICATE_NOENCRYPT, false);
// open a connection to the server
StreamConnection connection =
     (StreamConnection) Connector.open(url);
// Send/receive data
try {
     byte buf[] = new byte[200];
     String msg = "Test message";
     InputStream is = connection.openInputStream();
     OutputStream os = connection.openOutputStream();
     // send data to the server
     os.write(msg.getBytes);
     // read data from the server
     is.read(buf);
     connection.close();
} catch(IOException e) {
     e.printStackTrace();
}
```



NitroX™ Web Application Development for Eclipse

NitroX's AppXRay™ penetrates all layers of your web application and helps annihilate your web application development problems!

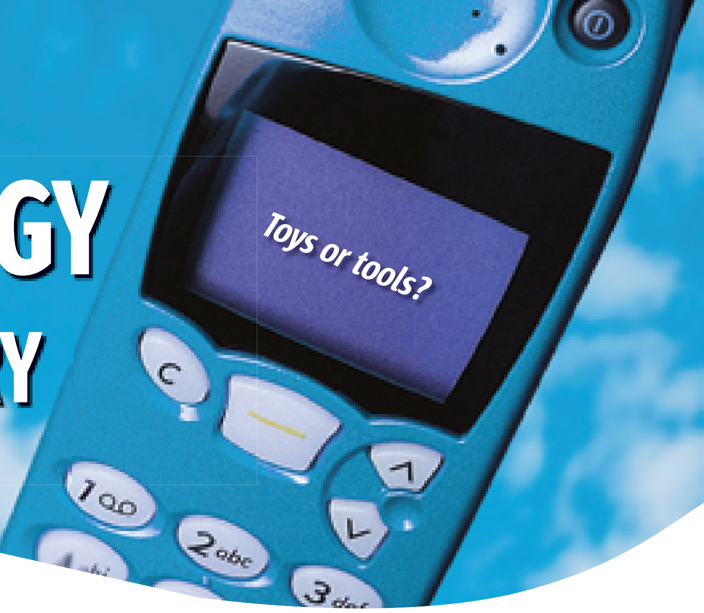
NitroX AppXRay unique features include:

- Debug JSP tags, Java scriplets, jsp: include, etc. directly within the JSP
- Advanced JSP 2.0 and JSTL support
- Advanced JSP editor – simultaneous 2-way source and visual editing with contextual code completion
- Real time consistency checking across all layers (JSP, Struts, and Java)
- Advanced Struts support – source and visual editors for Validation Framework, Tiles and Struts configuration
- AppXnavigator™ – extends Eclipse hyperlink style navigation to JSP and Struts
- AppXaminer™- analyze complex relationships between ALL web artifacts
- Immediate access to variables at all levels of the web application

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

M7®

JAVA TECHNOLOGY FOR THE WIRELESS INDUSTRY



by David Parsons, Ilan Kirsh, and Mark Cranshaw

The Java Technology for the Wireless Industry specification (JTWI) encompasses a standard set of J2ME APIs for mobile device development that is being widely adopted by mobile telephone service providers, making it an important platform for Java developers.

Its core component, the Mobile Information Device Profile (MIDP), provides a number of specialized libraries for multimedia and games development; however, its underlying subset of general purpose Java classes is strictly limited. In addition, support for persistence via the Record Management System is relatively poor. This raises an important question: Is JTWI a realistic application development tool or is it only good for games and other software trivia?

In this article, we try to answer this question by exploring the viability of MIDP as a tool for nontrivial application development. An enterprise application that includes mobile components might reasonably expect to devolve some of its business processes and data management to mobile devices. Our chosen example, which considers both of these aspects, is a proposed implementation of the Java Data Objects (JDO) specification. This includes a number of interesting features that highlight the constraints of working with J2ME APIs for limited devices. We describe the issues around the development of such an implementation and the limitations that MIDP imposes, suggest some useful workarounds and architectural options, and finally draw some conclusions about the usefulness of JTWI as a set of APIs for serious application development.

Introduction

Handheld computers, such as the Pocket PC and the Palm, can support reasonably complete Java Application Programming Interface (API) sets that can be used to develop serious enterprise applications. However, smaller devices, such as mobile telephones, only support Java APIs that have been significantly reduced to work within the confines of limited hardware. There is no support for the types of persistence mechanisms that we have come to expect on larger Java platforms. The question we address in this article is whether the mobile telephone is a viable platform yet for serious business or scientific applications that need to store and process data locally and that expect the services of a reasonably rich set of Java APIs.

To date, we have seen considerable development in areas such as mobile games development, but more serious business and scientific applications will need to be developed if JTWI is to be a useful component of enterprise software systems. Since such systems are likely to require considerable support for persistence, we focus on the JDO specification and examine some potential issues that arise when attempting to implement this specification using MIDP. We extrapolate from this analysis to assess the general usefulness of MIDP as a general purpose application programming platform. We also look at how MIDP devices fit into larger distributed architectures that can mitigate the limitations of mobile telephones as Java application platforms.

The Java 2 Micro Edition (J2ME)

To cater to a wide range of small devices and application requirements, the J2ME architecture (see Figure 1) provides multiple configuration and profile layers that overlay the specialized Java Virtual Machine (JVM) and operating system. Configurations define the minimum set of available JVM features and class libraries for a specific category of device and are hardware focused; profiles define the set of APIs available for a particular market category of devices and are software focused.

J2ME configurations specify the minimum requirements for memory, Java language features, JVM support, and runtime libraries. There are two standard J2ME configurations: the Connected Device Configuration (CDC) and the Connected Limited Device Configuration (CLDC). For the smallest portable devices, such as mobile telephones, the standard configuration is the CLDC. This configuration requires a very small virtual machine, such as Sun's KVM (Kilobyte Virtual Machine) or CLDC HotSpot Implementation, with footprints of only about 50–80K. These virtual machines don't have to comply with the full JVM specification, nor do they have to support the complete Java language specification. API support is limited to a selection of classes from a few packages from the Java 2 Standard Edition (J2SE), plus the Generic Connection Framework (GCF), comprising a hierarchy of connection interfaces (and the Connector factory class) that are intended to provide a generic way of expressing operations on connections regardless of the actual protocol.

David Parsons is a senior lecturer in information systems at Massey University, Auckland, and a knowledge engineer for Software Education Associates, Wellington. Until last year he was the director of emerging technologies at Valtech, London, and prior to that, principal technologist at BEA Systems.

d.p.parsons@massey.ac.nz

Java Technology for the Wireless Industry

The key goal of the JTWI specification is "to minimize API fragmentation in the mobile phone device market, and to deliver a predictable, clear specification for device manufacturers, operators, and application developers" (http://jcp.org/aboutJava/communityprocess/final/jsr185/index.html).

Thus, we can reasonably expect the next generation of Java-enabled telephones to support these technologies. The specifications included within JTWI are:

- ***Mandatory specifications:***
  –Mobile Information Device Protocol (MIDP) 2.0
  –Wireless Messaging API (WMA) 1.1
- ***Optional specification:***
  –Mobile Media API (MMAPI) 1.1
- ***Minimum configuration on which JTWI is built:***
  –Connected Limited Device Configuration (CLDC) 1.0

From an application development perspective, the most important API is MIDP (and by implication the CLDC upon which it builds), since these are the APIs that provide a subset of the standard Java packages found in the Java 2 Standard Edition, along with additional APIs specifically tailored for mobile development. One important issue with JTWI is that it mandates only CLDC 1.0, not CLDC 1.1, which, as we will see, introduced some important new features.

The Mobile Information Device Profile

The MIDP is one of two profiles (the other being the Information Module Profile) that works on top of the CLDC. It provides graphical interfaces for interactive applications and is the standard Java profile for mobile telephone development under the JTWI specification.

There are seven packages containing the additional classes and interfaces of the MIDP, providing user interface features at two levels of portability, sound support, certificate-based authentication, persistence, and the MIDlet framework for deploying classes into a MIDP environment. There are also extra classes in the javax.microedition.io and java.util packages.

The sum of APIs available to a MIDP developer will be the combined set of classes in the CLDC and MIDP. Figure 1 summarizes the relationships between the relevant configuration and profile APIs and the underlying J2ME JVM.

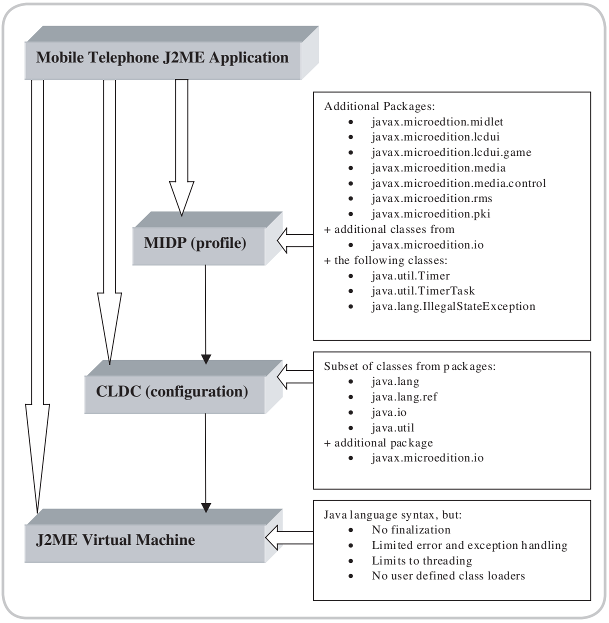


Figure 1 The APIs for MIDP development

What Is Missing from MIDP?

To consider how viable MIDP is as a general-purpose programming framework, it's useful to explore which packages and classes are excluded from it when compared with the standard edition. Of course, MIDP provides its own classes for graphics and sound, so there are no AWT, Swing, or sound-related packages from the standard edition. Similarly, MIDP has its own (limited) security classes, so there are no javax.security packages either. Since the Generic Connection Framework covers connectivity, packages relating to CORBA and network connections are also excluded. RMI likewise is not part of MIDP; although there is a separate J2ME RMI profile, it can only be used with the CDC configuration, not with CLDC. Other missing packages are those that relate to JavaBeans, reflection, XML, printing, and JNDI.

Although MIDP excludes many packages that are present in the standard edition, many of these have equivalents in the MIDP packages or, like printing, are not particularly important for mobile devices. However, it is in those packages that are included in MIDP that we find the most constraining factors, since these packages have far fewer classes in MIDP than in the standard edition. For example, MIDP includes only one interface and nine classes from java.util, as opposed to 14 interfaces and 41 classes in the standard edition (version 1.4), principally due to the absence of the Java 2 Collections Framework.

MIDP Persistence with RMS

In the context of enterprise Java development, there are a number of standard APIs that can be used to support data and/or object persistence: serialization, JDBC, Java Data Objects, and entity Enterprise JavaBeans (Enterprise Edition only). In contrast, MIDP does not automatically support any of these persistence mechanisms. The CLDC java.io package contains only the lower-level streams, readers, and writers, and doesn't contain any file or object streams, or, indeed, the Serializable interface.

This means that persistence-related code in MIDP differs considerably from other Java programming contexts and uses the Record Management System (RMS). The RMS comprises variable length record stores, each of which is a collection of variable size binary data records. Each record store has a unique name, and each record within a store has a non-reusable integer index that acts as a primary key. Although individual operations on a record store are atomic, there is no transactional support, apart from a version number, that can be used to support manually implemented locking strategies.

The Java Data Objects Specification

The Java Data Objects specification is an output of the Java Community Process (JSR 12), which had its first final release in April 2003. JDO defines an interface-based standard for the persistence of domain objects. There are currently around 20 vendors offering JDO implementations with varying levels of specification compliance. JDO implementations can be used with a range of data stores and across all three editions of the Java 2 platform and is a recommended persistence mechanism for data-centric applications on mobile devices.

JDO can be used in the context of a nonmanaged or a managed scenario. The former case refers to a typical two-tier or embedded application, the latter to a server-based architecture

Ilan Kirsh is a lecturer in Computer Science at the Academic College of Tel-Aviv-Yaffo, and the author of ObjectDB (www.objectdb.com), a pure JDO object database for J2EE, J2SE and J2ME.

kirsh@objectdb.com

Mark Cranshaw is a senior lecturer in the Faculty of Technology at Southampton Institute. He has extensive experience of delivering education using Java technologies, including JDO tools. His current research interests include mobile Java and HCI.

mark.cranshaw@solent.ac.uk

with the JDO implementation resident within a J2EE container. In both cases the JDO implementation hides issues specific to the Enterprise Information System (EIS), such as data type mapping, relationship mapping, and data retrieval and storage, from the application components. In addition, the managed scenario allows the application to make use of J2EE container mechanisms for transactions, security, and connection management. Clearly any JDO implementation within J2ME would be non-managed, though a distributed architecture that used JDO on mobile devices and also on a remote server would be possible.

The rationale behind JDO is that an application needs to be extended only once to encompass the JDO architecture and will then be able to access multiple and different EISs: object database systems, relational database systems, mainframe transaction processing systems, or ERP systems. The EIS vendor will be able to create a single JDO implementation and in doing so will allow pluggable access to any JDO-compliant application. The ability to run JDO in a MIDP context depends on the ability of the JDO implementation to provide the services required by the specification using only the resources available in a small device.

From the application perspective, the primary interface is PersistenceManager, supported by the Query and Transaction interfaces. An object implementing this interface will be responsible for the management of the JDO instance life cycle. This includes reading and writing from the data source and working with Query and Transaction objects to create the illusion that the entire network of objects reachable from the application, including persistent objects, are resident in memory at the same time. Within a nonmanaged environment, a JDO application will retrieve a PersistenceManager directly from a PersistenceManagerFactory instance provided by the JDO implementation.

Candidate persistent classes must implement the PersistenceCapable interface, typically through the use of an enhancement tool provided as part of the JDO implementation. Enhancement is carried out after the domain model is complete and is usually applied to the compiled byte code, making this a completely transparent process. The binary compatibility of JDO instances means that they are portable between EISs without recompilation, provided a JDO implementation is available. It would not be realistic to expect a mobile JDO implementation to include the development tools, such as the byte code enhancer, for JDO. Mobile devices are unlikely to be used as software development platforms. Rather, the device would be required to act as a JDO client, downloading pre-enhanced byte code and using the JDO client APIs. However, it's not enough for a JDO client application to have only byte code, since any class that is to be persisted must have a corresponding entry in an XML file, known as the persistence descriptor file. The developer may supply a separate descriptor file for each class or a single file for a package. This file, containing key mapping information, must be available to the JDO implementation at runtime. Therefore, a MIDP JDO implementation would need to include both the JDO client APIs and the means to process the persistence descriptor.

Implementation Issues for JDO on MIDP

Current JDO implementations cannot run on highly constrained devices using the J2ME CLDC configuration. The smallest realistic JDO platform is currently a CDC configuration using a larger nonstandard library, such as the IBM JCL Max available with the J9 VM. This kind of configuration is appropriate for Pocket PC type devices but not for mobile telephones.

There are a number of reasons why implementing JDO in a CLDC/MIDP environment is problematical. The core issue is the required footprint size, because a JDO implementation would require space for the implementation itself, plus the larger byte code of classes enhanced to become persistent. Object data would have to be stored using RMS or a proprietary JDBC implementation. Depending on the way that it mapped the objects, overall storage space requirements for JDO persistence would probably be larger than the most optimized equivalent using direct RMS. Another problem would be the speed of RMS, since it's just a set of indexed flat files that require either direct indexed access or sequential access to locate data. This could be very inefficient when attempting to implement the JDO query language, JDOQL.

As we've discussed, JDO also requires an XML persistence descriptor that is accessible at runtime. A full validating parser to process this file would be much too large, but there are highly optimized nonvalidating XML parsers designed for use with J2ME, such as kXML or others that implement the XML APIs of the J2ME Web Services Specification. An alternative strategy might be to use some other way of providing metadata; for example, using the optional JAD (application descriptor) or manifest files.

Another limitation of the MIDP APIs is that they don't include the JDK 1.2 collections framework. This means that the only container classes available are the Vector and Hashtable from 1.x versions of Java, which is inconsistent with the JDO requirements of at least supporting the HashSet. We could develop some kind of wrapper class around the Vector to produce a different type of container that can be used in a J2ME context, but this doesn't address the further issue that Vector is synchronized, which could impact performance. Another option is to write a custom HashSet class (using an array of linked lists) and/or other custom library classes.

To mitigate the potentially large footprint resulting from the proliferation of the additional classes necessary to make JDO function in a MIDP context, a byte code shrinkage and/or obfuscation tool could be used. Since the main shrinkage benefit is gained where only small parts of libraries are used, the already small J2ME libraries would not necessarily yield a major change in footprint. For example, test results show only a 16% reduction in size for a sample J2ME application, as opposed to a best result of 91%.

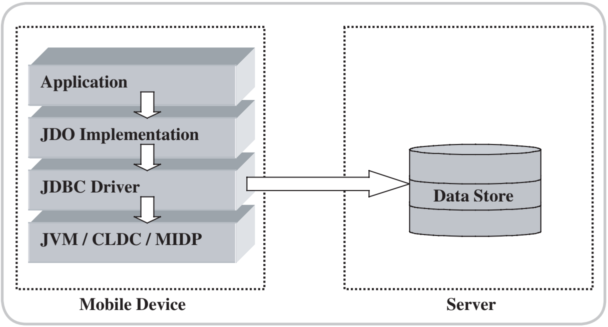


Figure 2 Client JDO architecture for MIDP

In addition to limitations of the MIDP APIs, the JTWI specification imposes further restrictions by mandating only version 1.0 of CLDC. This means that JDO implementations based on CLDC 1.1 could not be guaranteed to work on a device that was JTWI compliant. One significant issue here is the lack of floating point support, which is required by JDO but only supported in CLDC from version 1.1.

Similarly, options for providing the required support for the cache are restricted. Implementing a JDO-compatible cache requires weak or soft references, because the JDO specification requires that persistent objects remain in the cache as long as they are in use by the application, but that they be removed from the cache automatically when the application stops using them. Because the application does not report when objects are disposed of, the only way for a JDO application to know that an object can be removed from the cache is to use weak or soft references.

CLDC does not include support for soft references and weak references have only been supported since version 1.1. An alternative approach for managing the cache could be to hold objects in the cache only when a transaction is active. When the application reports the transaction closed, all the objects that were retrieved during that transaction could be removed from the cache. This approach was common in object database implementations for older Java versions, in which weak or soft references were not provided.

Given these constraints, a JDO implementation for MIDP cannot easily meet the full JDO specification and pass the Technology Compatibility Kit (TDK) tests. It would be realistic to follow the model adopted by Oracle TopLink of supporting the JDO APIs as closely as possible within the constraints imposed. Braig and Gemkow, in their article "The BonSai Principle," demonstrate a similarly cut-down implementation, supporting only parts of the API (e.g., no query language) based on AspectJ. However, their implementation does at least run within the confines of a MIDP environment.

It may be that the best that can be achieved given current constraints is an architecture where remote proxies are used in combination with the JDO implementation running on the server. This should not be seen as a purely negative situation, since any enterprise-level application is likely to require distributed access to applications and data that would not benefit from very heavyweight mobile clients. Rather, the bulk of the system's services and data would be server-centric, with only small subprocesses and data caches being devolved down to individual devices. In this type of architecture, a JDO implementation could encapsulate the distributed cache management required and assist in the transparency of developing distributed applications.

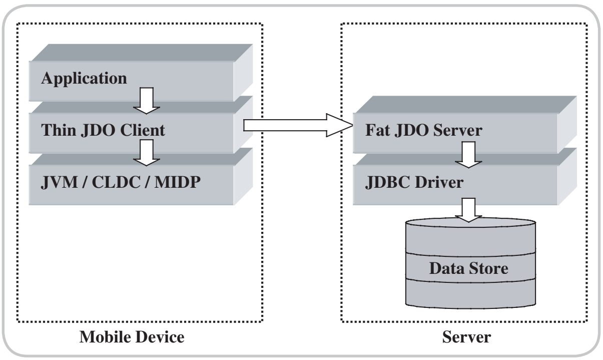


Figure 3 Client/server JDO architecture for MIDP

JDO Architectures for MIDP Devices

A MIDP application can benefit from using JDO in two different scenarios. In the first scenario, the data store is located on a central server and a MIDP application running on a client mobile device uses JDO to access the remote data. The JDO layer in this case functions as a high-level wrapper for communication with the remote server's data store. In the second scenario, JDO is used to manage local data on the mobile device. The JDO layer in this case functions as a high-level wrapper around the MIDP RMS mechanism. In both scenarios, JDO provides a similar, easy-to-use API for managing the data, whether local or remote, using the application domain model. Each scenario, however, requires a different design solution.

Client JDO Architecture

A common technique in the design of JDO implementations is to build a JDO-JDBC bridge. An implementation that functions as a wrapper layer around JDBC can easily support a wide range of data sources. Figure 2 shows a naive architecture for using a client JDO implementation of MIDP based on this approach.

In this architecture, the data store is managed by a remote process that runs on a central server. A JDBC driver on the mobile device communicates with the remote server process, while the JDO implementation functions as a layer between the MIDP application and the JDBC driver.

We've got problems with your name on them.

At Google, we process the world's information and make it accessible to the world's population. As you might imagine, this task poses considerable challenges. Maybe you can help.

We're looking for experienced software engineers with superb design and implementation skills and expertise in the following areas:

- high-performance distributed systems
- operating systems
- data mining
- information retrieval
- machine learning
- and/or related areas

If you have a proven track record based on cutting-edge research and/or large-scale systems development in these areas, we have brain-bursting projects with your name on them in Mountain View, Santa Monica, New York, Bangalore, Hyderabad, Zurich and Tokyo.

Ready for the challenge of a lifetime?
Visit us at http://www.google.com/jdj for information. EOE

Google

This architecture has several disadvantages. First, because the entire JDO implementation, including the JDBC driver, is running on the mobile device, substantial memory resources are required. Second, standard JDO operations, such as converting queries from JDOQL to SQL (i.e., from the query language of JDO to a format that a JDBC driver can handle), might be too slow on the standard CPU of a mobile device. A further issue is the limited number of JDBC drivers that are available today for MIDP.

Client/Server JDO Architecture

Because of the problems associated with deploying the entire JDO implementation on the mobile client, a more realistic architecture to develop would be a client/server JDO framework (see Figure 3). In this architecture the JDO implementation is split between the mobile device and the server. A "fat" JDO process runs on the central server, communicating with the data store using a JDBC driver.

On the mobile device, the MIDP application uses a thin JDO client library to access the data store. Only operations that must be implemented on the client side are included in the thin JDO client library; all other operations are implemented by the JDO server. For instance, converting queries from JDOQL to SQL should be done on the server side due to the limited resources, in terms of memory size and CPU speed, of the mobile device.

To keep the footprint of the client as small as possible, some features of JDO, such as supporting local queries on memory collections (which, if supported, would have to be implemented on the client), may be omitted. This would mean the loss of full JDO compatibility.

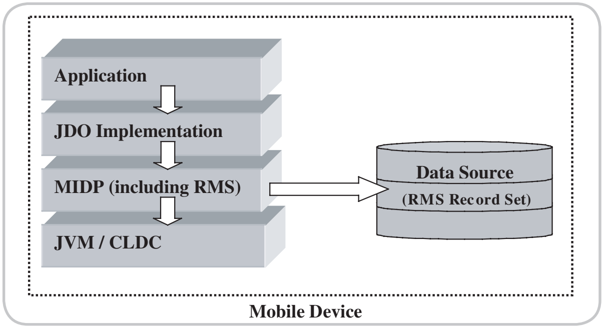


Figure 4 Local storage JDO architecture for MIDP

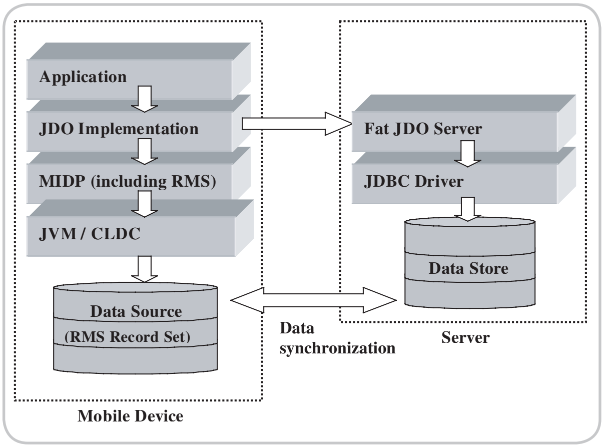


Figure 5 Hybrid architecture

The client/server JDO architecture is very flexible and can be tailored according to requirements. For example, instead of using a two-tier architecture, in which the data store and the JDO server are located on the same machine, a three-tier architecture can be used, deploying them on two different machines. Another design change is required when a JDBC driver is not available, for example, if the data store is an object database. In that case, the JDO server is expected to access the data store directly.

Local Storage JDO Architecture

There are many benefits to storing the application data on a central server, but in some situations local storage may be preferred. For example, a MIDP application that manages a contact list or a personal organizer should keep the data locally on the mobile device (possibly in addition to a backup of that data on a remote server). This can provide a faster response time and also ensures the availability of data when the server is unreachable because of network problems or maintenance. However, the JDO implementation, which acts as a wrapper around the RMS APIs, is complex (see Figure 4).

To understand what is expected from local storage JDO for MIDP, it might be helpful to distinguish between two types of JDO implementations: those that provide JDO support for relational databases by implementing a JDO-JDBC bridge and those that use object databases to support the JDO APIs. Because object databases do not rely on another database system as back-end storage, they must provide all the services that a standard database system provides, including (among others) storage management, lock management, query processing, and transaction support. For example, object database JDO implementations cannot convert JDOQL into an SQL query that is executed by a relational database, but rather have to include their own mechanism for processing and executing queries.

A local storage JDO implementation for MIDP would be very similar to a JDO object database implementation. Such an implementation has to support all the standard services that a database provides and everything has to be implemented on the mobile device. RMS as a low-level storage system is at least as good as binary files, but does not provide the database services that JDBC provides.

A storage solution can be based on allocating the first record in the RMS record set for general database information and an additional record for every object and every class schema. Other common internal database data structures, such as BTree+ for indexes, can be implemented by multiple RMS records (for instance, every node in the tree could be stored in a record). We have to consider whether or not the local management of such data structures is realistic because of the memory resources that they consume on the device, both for storing the data structures and in the implementation byte code that they add. A more appropriate solution might be to avoid supporting indexes (which are not required by the JDO specification) and to process queries by iteration over all the objects one by one (using RMS filters).

Hybrid Architecture

The most flexible solution for JDO on MIDP would be a hybrid architecture that included elements of both the client/server and local storage approaches. This would provide the benefits of a fully featured JDO implementation running in the server, plus the ability to preserve disconnected local data to maintain quality of service (see Figure 5). Of course, this type of solution is much more complex, since it requires the JDO implementation to manage the distributed data that is being cached on mobile devices. We can expect the development of systems using this kind of architecture to be supported by implementations of the Java synchronization APIs.

Conclusion

The current version of the MIDP specification is an interim set of APIs that reflects a particular point in the development of mobile telephone technology. At present, mobile phone developers must work within the constraints of current devices and work around the constraints of the platform as best they can. Although the limited CLDC/MIDP libraries constrain multiple aspects of Java application development, there are a number of initiatives in place to support applications migrating down to smaller devices, including small footprint XML parsers and databases.

Regarding JDO and whether or not it could be implemented to run on a MIDP device, our conclusion is that while MIDP alone cannot realistically host a full JDO implementation, a distributed implementation that combines local processing with server support can indeed meet our application needs. Not only that, but such an architecture actually opens up a more challenging set of options for truly distributed systems that provide for widely distributed data and processes.

The real challenge for MIDP developers is to build applications that not only work locally on a single device but can interact and synchronize with multiple nodes of different types in a disparate architecture. In practice, running JDO on a single device provides few advantages over alternative APIs for data access. However, a distributed JDO implementation that integrated and synchronized data across multiple nodes, encapsulated behind a single distributed object model, could be a very valuable tool.

From our discussion of JDO as an example of serious application development, we can see that developing software for mobilized architectures requires us to consider a range of aspects of design and implementation in order to identify the optimum configuration. MIDP alone cannot provide a fully featured Java deployment platform. But, by playing to its strengths, such as the ability to maintain a persistent local data cache and support it with server-side resources, it opens up a range of new opportunities in software development.

References

- Kochnev, D., and Terekhov, A. “Surviving Java for mobiles.” *IEEE Pervasive Computing*, Vol. 2, no.2, June 2003.
- Sun Microsystems. (2003). “JSR-000185 Java Technology for the Wireless Industry 1.0 (Final Release).” Java Community Process: http://jcp.org/aboutJava/communityprocess/final/jsr185/index.html
- *JDOCentral, Developer’s Community for Java Data Objects:* www.JDOCentral.com
- Reese, G. (2003). *Java Database Best Practices*. O’Reilly.
- *ProGuard Java Class File Shrinker and Obfuscator. Test results:* http://proguard.sourceforge.net
- *Java Data Objects Expert Group, “JSR-000012 Java Data Objects 1.0.1 (Maintenance Release), 2003, section 5.5.4.” Java Community Process:* http://jcp.org/aboutJava/communityprocess/final/jsr012/index2.html
- Dubé, J.; Sapir, R.; Purich, P.; and Siegal, E. “Oracle Application Server TopLink - Application Developer’s Guide 10g (9.0.4).” Oracle Technology Network, pp. 470–486, 2003: http://download-west.oracle.com/docs/cd/B10464_01/web.904/b10313.pdf
- Braig A., and Gemkow, S. “The BonSai Principle – Persistenz in der Java 2 Micro Edition.” *Java Spektrum*. September 2002: www.sigs.de/publications/js/2002/09/Braig_JS_09_02.pdf

From the Group Publisher

–continued from page 3

- *Ken Iverson:* Inventor of APL, later J
- *William Kahan:* “The Old Man of Floating-Point”; primary architect behind the IEEE 754 standard for floating-point computation
- *Robert Kahn:* Co-inventor of the first Internetworking protocol, TCP
- *Mike Karels:* System architect for 4.3BSD
- *Alan Kay:* Inventor of SmallTalk
- *Gary Kildall:* Author of the archetypical OS known as CP/M (control program for microcomputers)
- *Donald Knuth:* “Father of Computer Science”; author of *The Art of Computer Programming*; inventor of TeX, allowing typesetting of text and mathematical formulas on a PC
- *Butler Lampson:* Architect of Cedar/Mesa; implementer of Xerox Alto
- *Robert C. Martin:* Agile software development proponent; CEO, president, and founder of Object Mentor
- *Yukihiro Matsumoto (“Matz”):* Creator of Ruby
- *John McCarthy:* Creator, with his graduate students, of Lisp
- *Doug McIlroy:* Head of department at Bell Labs where Unix started
- *Bob Metcalfe:* Creator of Ethernet
- *Chuck Moore:* Inventor of Forth, a high-level programming language
- *Andrew Morton:* Linus’ No. 2 in the kernel group
- *Ted Nelson:* Creator of the Xanadu project – universal, democratic hypertext library; precursor to the WWW
- *Kristen Nygaard:* Developer (with Ole-Johan Dahl) of SIMULA, the first object-oriented programming language
- *Peter Pag:* Pioneer of 4GLS (1979); developed Software AG’s Natural
- *Bob Pasker:* Founder of WebLogic, author of the first Java application server
- *Benjamin Pierce:* Harvard University faculty member for 49 years; recognized in his time as one of America’s leading mathematicians
- *P.J. Plauger:* Chair of the ANSI C committee
- *Jon Postel:* “The ‘North Star’ Who Defined the Internet”
- *John Postley:* Developed Mark IV (1967), the first million dollar software product, for Informatics
- *Martin Richards:* Designer of the BCPL Cintcode System
- *Martin Roesch:* Author of the open source program Snort, in 1998
- *Gurusamy Sarathy:* Heavily involved in maintaining the mainstream releases of Perl for the past seven years
- *Carl Sassenrath:* Author of REBOL, a scripting language
- *Guy L. Steele:* Author of authoritative books and papers on Lisp
- *W. Richard Stevens:* “Guru of the Unix Gurus”; author and consultant
- *Ivan Sutherland:* Considered by many to be the creator of Computer Graphics
- *Avadis (Avie) Tevanian:* Chief software technology officer, Apple
- *Guy (Bud) Tribble:* One of the industry’s top experts in software design and object-oriented programming
- *Patrick Volkerding:* Creator of Slackware Linux
- *John von Neumann:* Pioneer of logical design; first computer theorist to tackle the problem of obtaining reliable answers from a machine with unreliable components
- *Larry Wall:* Author of Perl
- *John Warnock:* Inventor of PostScript; CEO of Adobe Systems
- *Michael “Monty” Widenius:* Creator of MySQL
- *Nicklaus Wirth:* Inventor of Algol W, Pascal, Modula, Modula-2, and Oberon
- *Stephen Wolfram:* Scientist, creator of Mathematica
- *Jamie Zawinski:* Instrumental in the creation of Lucid Emacs (now XEmacs)

Calvin Austin
Core and Internals Editor

Java Inside

I recently upgraded my home network to use 802.11g. The prices for routers and PC cards have fallen throughout the year, but unfortunately the support has only marginally improved in that time and, after many re-installs, my network wasn't working. I eventually had to resort to probing the card to find out the exact chipset to get my Linux box up and running. It's fine now, but I found it ironic that probably the most important part of the card that would ensure compatibility was completely hidden, both on the box and in all of the documentation.

Things are a little different in embedded Java land. As a core Java developer you've probably heard of J2ME and technologies like MIDP and you could claim that the Java runtime inside a J2EE application server is at least hidden, if not embedded. Unlike my network card, with phones and other devices it's very clear whether they use Java technology and there's a very good reason for that.

I don't get to spend that much time with J2ME so I thought it would be useful to share a quick primer on J2ME technologies this month as a refresher. J2ME is a specification driven through the Java Community Process, just as J2EE and J2SE are specifications for the desktop and server platforms. As you would expect, devices typically have a set of constraints. Some have become more flexible over the last few years with improvements to CPU speed and also to the amount of memory available; others such as scheduling and process priorities have remained core fundamentals. The most discussed technologies that you may run into are CLDC, CDC, MIDP, and JTWI. The J2ME specification itself replaced Personal Java, which was a one-size-fits-all edition for small devices.

CLDC, Connected Limited Device Configuration, is a specification for very small footprint systems. Connected means that it can use Java networking APIs; limited refers to the footprint or resources. The configuration includes only a very small set of classes and reduced JVM. The first reference implementation that Sun released to adhere to that specification was called KVM. Adaptive compiler hotspot JVMs are now available for CLDC. Most of the popular deployments of CLDC are combined with an additional set of libraries called MIDP. The Mobile Information Device Profile provides the user interface classes, network management, security framework, media and games APIs, and more. This combination of CLDC and MIDP is the technology used in most Java-enabled phones.

The CDC, Connected Device Configuration, follows the same naming conventions as CLDC, but is for devices that can accommodate a larger footprint. Still designed for machines with as little as 2MB memory, this is significantly larger than CLDC, which scales down to 128KB. Again, there is a hotspot JVM implementation and the larger footprint size means that more of the J2SE APIs are supported. CDC also requires the use of profiles to deliver functionality, and with the personal profile you can support basic applets.

I've only mentioned a few profiles and have not even begun to discuss the optional packages that are available. Due to the rapid growth of packages and to ensure a basic level of functionality that extends beyond CLDC/MIDP, the Java Technology for the Wireless Industry (JWTI) specification was created. This in many ways is like a platform specification but defines the core level of functionality for devices that are JTWI compliant.

At this point you may be thinking what can I do with those configurations? They look like they're for device makers, not regular developers. That is partly correct, but you need a CDC or CLDC reference configuration to be able to run your wireless or embedded applications, and implementations of CDC and CLDC are available for running outside of a device on a regular PC. If you are interested in writing J2ME programs, the Java Wireless Toolkit is the place to start. This environment contains the core APIs you need to make your own phone application and to take advantage of the media APIs, wireless messaging, and the latest Bluetooth technology.

Sometimes knowing what's inside does make a difference. I would recommend giving the wireless toolkit a try; perhaps you have the next killer phone app in your reach!

Finally I wanted to mention one of the new developments in J2SE. Both the J2SE 5.0 source and now weekly binaries and source from J2SE 6.0 are available from http://java.sun.com. The J2SE 6.0 development is still in its early stages so if you are expecting to see big changes there, you'll need to wait a little while.

"Due to the rapid growth of packages and to ensure a basic level of functionality that extends beyond CLDC/MIDP, the JWTI specification was created"

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

calvin.austin@sys-con.com

If your application is Java, your database should be PointBase

Every day, millions of people manage their data with DataMirror's PointBase Embedded

Now you can deploy a full-featured database system that doesn't require OS-specific packaging and testing. PointBase® lets you experience the unmatched platform independence of Java™—all from a cost-effective, small footprint, Java RDBMS.

PointBase Embedded® is a pure Java relational database system that supports full SQL functionality and JDBC. Designed to run without manual administration, PointBase Embedded ensures data integrity and saves you development time and money.

PLUS, you can experience the many benefits that a pure Java environment offers, including cross-platform deployment, network security, and speed.

Leverage the speed and reliability of PointBase technology today! **DOWNLOAD** a FREE 30-day trial version at **www.pointbase.com/jdjjan05** and find out how companies are benefiting from the world's leading Java database.

Contact Us
408.961.1100 / 877.238.8798
sales@pointbase.com
www.pointbase.com

PointBase® a Division of DataMirror®

TRANSLATION-BASED INTEGRATION

Rocky road made smooth

by Alexander Krapf

Anyone who regularly works with more than one development language and a third-party library has faced the situation described by: "Great library, if only I could have it in my programming language." Some vendors make a living from publishing different language versions of their product, but many can't afford or don't want to pay the costs of maintaining several parallel implementations of their product.

Consequently, over the years many integration approaches have been created to help us with the common problem of using software written in one language in another language. Microsoft has even gone as far as building language inter-operability into the core of its .NET platform (of course, they excluded Java). When talking about .NET, it should also be mentioned that a .NET-compatible language has to satisfy many constraints imposed by the .NET platform. So many constraints, in fact, that some people have said you don't really write your code in C++ or VB, but rather in a particular dialect of .NET. For anybody who is not using .NET, or who has to integrate .NET with .NET-foreign languages such as Java, here are the basic integration approaches:

1. ***Source code translation:*** Translate the Java source code of a library to C++.
2. ***Byte code/binary code translation:*** Take a compiled Java class file and compile it into object code. This, as well as the previous approach, can also be referred to as "implementation translation," because the method bodies are converted.
3. ***In-process wrapping:*** Create a C++ wrapper for a Java type that internally uses JNI to delegate from C++ to Java.
4. ***Out-of-process wrapping (remoting):*** Make a Java type available to a C++ programmer by calling from the C++ application into a Java process via sockets. This, as well as the previous approach, can also be referred to as "interface translation," because only the API signatures are converted.
5. ***RPC/Web services/messaging:*** This is really a special case of (4) and involves the sending of messages between applications written in different languages. SOAP, CORBA, XML RPC, etc., are examples of this integration approach.

All of these different integration approaches have one thing in common – to work for a reasonably large code library or API, an automated translation/transformation engine is an absolute must. Imagine having to translate the Java runtime library to C++ by hand or having to hand-assemble and dispatch a SOAP request in a C++ program just to call a Java method.

Also, let's not forget that this type of integration problem does not start out with a portable interface description like CORBA's IDL or Web services' WSDL, but rather with a full, preexisting implementation in one of many possible technologies.

Let's take a quick look at some factors that influence our choice of integration technology before we delve into the more technical aspects. I haven't listed these points by importance; which points are important to you depends entirely on your specific integration project.

Granularity

At what level do we integrate? If we are dealing with a component API, i.e., an API that has relatively few integration points for relatively hefty operations, an RPC-based integration could be a good fit.

Performance

If we need fine-grained integration at the API level, any RPC-based solution is usually ruled out because we cross the language boundary too often for too little work, resulting in bad performance.

Reliability

Any out-of-process/RPC solution introduces new failure modes into an overall solution. Can you create a reliable integration solution for the specific API that you wish to use? If the answer is yes, how much more complicated is it going to be?

Security

Is an out-of-process/RPC solution without additional security mechanisms acceptable for your integration problem? Is additional security available and how much is it going to complicate the project or increase the price?

Alexander Krapf is president and cofounder of CodeMesh, Inc. Krapf has over 15 years of experience in software engineering, product development, and project management in the United States and Europe. He has been extensively involved in a variety of complex product development efforts, using his in-depth understanding of .NET, C++, and Java. Alexander has been published in technology journals and has been a speaker at a variety of industry conferences. He received a bachelor of science degree in electrical engineering from the University of Stuttgart, Germany.

alex@codemesh.com

Scalability

In-process integration solutions usually have the best performance by far, but they may be wasteful in terms of system resources, and scalability might benefit from being able to distribute integration components. On the other hand, some APIs cannot be distributed/remoted without jeopardizing reliability.

Fidelity

How true to the original should the integration API be? Should it be self-evident to a user of the original API or are some special skills (like JNI, CORBA, or SOAP) required?

Purity

Is it necessary to completely remove the original technology from the picture (i.e., by translating/porting the code) or can we bridge between the two sides? Bridging might be the only feasible solution if you don't have access to all the referenced pieces (for example, a source code translator does not work if the code that needs to be translated references a third-party library for which you don't have the source code).

Licensing

This is often an issue for translation-based tools: even if you can translate your own code for an integration solution, do you have the necessary rights to translate referenced source code (for example, from a J2SE source distribution)?

Maintenance

Is your integration solution tied to a specific version of a technology and how easy is it to regenerate the integration solution and upgrade to a new version of the technology?

Clearly, given an API in one language, we would love to have the following: an automatically generated, highly reliable, totally secure, highly performing, true-to-the-original, semantically equivalent API in the other language. World peace would be another nice thing to have, while we're at it.

In the real world, of course, things are not that simple. I always like using Java and C++ as an example – first, because I know a lot about these two languages and, second, because they are deceptively similar. Many people think that it should be simple to translate an application written in one language into the other language. Let's take a closer look at this. The Java class in the following code will be our first example.

```
public class Person
{
  public String       name;
  public Person() { name = null; }
  public Person( Person p ) { name = p.name; }
  public String getName() { return name; }
  public void   setName( String n ) { name = n; }
  public static void  delete( Person p ) { PersonUtil.delete(
p.name );
  public static Person create( String n ) { return new Person( n
); }
}
```

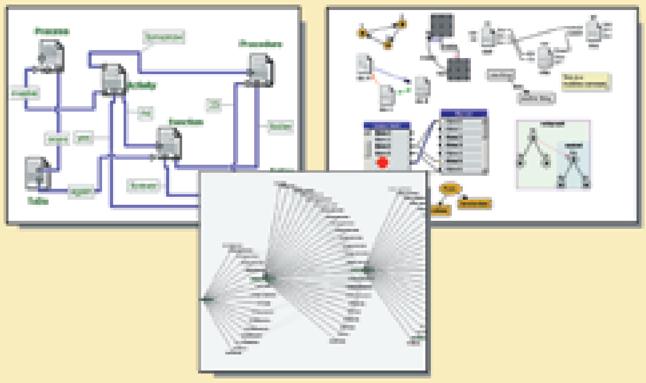
This is a fairly straightforward Java class. The only slightly unusual thing about it is its public data member called name. Maybe we were forced to make it public because we're using this type with a JavaSpace or we might simply have made it public due to a design oversight.

Now let's cook up a C++ class that does the same thing. That should be easy, shouldn't it? We could probably write a simple parser and translator and come up with the automatically generated C++ type in the following.

```
class Person
{
public:
  std::string       name;
  Person() : name() {}
  Person( const Person & p ) : name( p.name ) {}
  ~Person() {}
  std::string       getName() { return name; }
  void               setName( std::string n ) { name = n }
  static void       delete( Person & p )
  {
    PersonUtil::delete( p.name );
  }
  static Person     create( std::string n )
  {
    return Person( n );
  }
};
```

That's a straightforward look-alike of the original Java type in C++. We'll have at least one immediately obvious problem here: delete is a reserved word in C++, so the generated code won't compile. In the case of delete this is an easily solved problem; we simply use _delete or some

Build Incredible Interactive Diagrams
with JGo™



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

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

Northwoods Software Corporation

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

other mangled form of the reserved word. The ease of solving this problem hides a deeper problem though: C++ has a preprocessor that usually predefines many macros. Each of those macros really has to be treated like a reserved word, and that is going to be complicated without a sophisticated tool.

Then there is a host of less obvious problems with this naïve translation. Let's go through them one by one.

1. We translated the Java Person(Person) constructor into the C++ copy constructor. This is totally appropriate based on its call signature, but it might not be appropriate semantically. The C++ copy constructor is invoked automatically by the compiler in many scenarios, whereas the original Java constructor is only supposed to be invoked explicitly by the programmer. Will our generated code have the correct semantics? It depends on the type, which is usually not a satisfactory answer.
2. In Java, we had a string field; in C++ we have a corresponding std::string field. There's another semantic difference hidden in this seemingly obvious translation: String instances are immutable, whereas std::string instances can be modified. Yes, we could try working with const or mutable modifiers, but those modifiers might collide with other intended semantic usages.
3. The create method in the Java class returns a new Person instance. We're facing the question of what to return from the corresponding C++ method. If we return the Person instance by value as in the sample code, we make sure that it does not get created on the heap and that we don't have a memory leak if we ignore the return value (something that's usually safe to do in garbage collecting Java). The downside of this approach is that we end up with an instance of class Person. We can now never return a subtype of Person from this method unless we return the result through a pointer to Person or a reference to Person. But if we return the result via a pointer, we put the burden of freeing the instance on the caller, which is a built-in memory leak as the return value is usually ignored on the Java side. This demonstrates how a perfectly correct and proper translation can yield either semantically incorrect or semantically hard-to-use code. It's a true catch 22.
4. In the delete method, we're calling into the PersonUtil type. This means we have to translate this as well, or make it available in some other fashion in order for the integration project to be a success. In a typical integration project, you quickly face a type explosion due to these dependencies. For example, if you analyze the Java type object, you find out that it references (directly or indirectly) between 250 and 350 types. A good translation-based integration tool will allow you to prune down the type set to just the types that you're interested in without compromising usability.

Other big semantic issues in a Java/C++ translation involve:

- Inheritance semantics (is inheritance allowed or not)
- Exception semantics (C++ exception declarations have very different semantics from Java exception declarations)
- Interface semantics
- Life cycle management and copy semantics
- Thread semantics

Just translating a Java statement to a corresponding C++ statement rarely does the job. The problem is mostly not the syntax differences between the two languages but rather the semantic differences between the languages. Also think about the work that might be required to replace the platform infrastructure like the String class. In Java, String has a lot of functionality; how much depends on which version of String you're looking at. If we simply translate String to std::string, what's going to happen to the original Java functionality of String?

The point I'm trying to make is that naïve translations very quickly run into lots of problems unless the translated API is very small. Where does this leave us? It leaves us with two basic options.

1. *Put the semantic integration burden on the user*

This is the path that most EAI strategies like Web services or CORBA take. The user creates what amounts to an integration model (IDL, WSDL) and implements it in terms of his or her favorite technology.

Some tools can help by generating integration models based on existing code, but, as I already pointed out, this is a tricky proposition because some semantic usability (extensibility, life-cycle, etc.) cannot be expressed in your integration model of choice. Also as already mentioned, this approach works best for component models with relatively few and relatively simple entry points. Simple means that you don't have many complex, user-defined or platform types, but rather primitive types or pseudo-primitive types, like String or Date.

2. *Put the semantic integration burden on the translation tool*

This is the path that is taken by some integration solutions that are exclusively targeting language-integration problems. These tools create very sophisticated type models that mirror (to the extent possible) the underlying type model.

The developer can use arbitrarily complex user- or platform-defined types as if they were written in his or her language.

In my opinion, the latter alternative is the better one because it avoids the pitfalls of having to constantly maintain an integration model and because the "unexpected" API differences are small. The more a developer has to learn when using a type that is written in a different language, the less successful the integration technology is going to be. Any special knowledge that is required creates "friction," and friction causes bugs, and bugs cost money and cause the tool to be unpopular.

I believe that the automated translation of code (source or object) into another language, be it an implementation translation or a calling interface translation, is the way to go for language integration.

LINUXWORLD CONFERENCE & EXPO

CONFERENCE: February 14 – 17, 2005 EXPO: February 15 – 17, 2005

HYNES CONVENTION CENTER • BOSTON, MA

Linux. Reliability. Efficiency. Cost Savings. If you're thinking about Linux or Open Source, the time is now. And LinuxWorld Conference & Expo is where you need to be. Expect the following cutting edge insights:

- Forward-thinking Keynote presentations by industry thought leaders from Novell, HP, Computer Associates, and MySQL.
- 10 hours of **FREE** educational programming on everything from the State of Linux and Open Source technology to Linux and desktop users.
- The world's leading hardware and software vendors as well as the hottest new start-ups giving you the opportunity to evaluate and test drive the latest technology.

And that's only the beginning. LinuxWorld Conference & Expo – where professionals come to do business better.

Register today at linuxworldexpo.com for your FREE Exhibit Hall pass.

LINUXWORLDEXPO.COM

Where OPEN MINDS Meet

Register Online With Priority Code: D2401

PLATINUM SPONSORS

BakBone Redefining Data Protection · ca Computer Associates · hp invent · IBM · Novell · ORACLE · Sun microsystems

IDG WORLD EXPO

Joe Winchester
Desktop Java Editor

The Return of the Pig

The key to building a distributed application successfully lies in a sensible partition of work across the different boundaries and devices. With a client/server program, one of the advantages it offers over a more traditional thin client is that for each task, instead of having to wait for the server to page the application back into memory, process the results of the display buffer, and prepare output, the PC is able to offload some of the validation and processing locally.

Not only is this more responsive to the user, but it makes sense to have a physical division of responsibilities in which code logic is executed closest to where relevant resources lie. Field-level validation, defaulting, and completion of values can all be done on the client. Even where such processing requires trips to the server, an atomic call to the server on a text field's focus-change event can easily validate an entry against a database. By scheduling the display thread this way the GUI can remain responsive. Another benefit of using the client's windowing subsystem fully is the ability to open multiple shells or dialogs simultaneously and have the user move, size, and arrange them to make the best use of the available space.

In a presentation to users or a sales pitch to potential customers, the eye-candy of a windows program will always win over a terminal-based application. In chapter one of the client/server wars, those with a lot of investment in back-end technology required a quick fix to counter the glitterati of the GUI, and one technique that arose is "screen scraping." This is where the data stream intended for the server's terminal is interpreted and re-presented on the desktop using a best guess set of controls and widgets. From the back of a dimmed room in a demo or on the noisy floor of a software booth at a conference, it looks pretty impressive. It's no more than a fool's shiny silver bullet, however, as all that it gains is the glitter of the GUI without any accompanying depth. The transactional mode of the application remains with logic being done on the server; simultaneous multiple windows don't occur as the workflow is restricted to merely coloring in the terminal's display, and an extra layer of processing has been added to the previously working program for very little perceived benefit.

One metaphor often used to describe screen scraping is "lipstick on a pig," perhaps loaded slightly unfairly with the implication being that the original application shares its characteristics with a snorting, mud-loving suidae beast. What is correct with the analogy is that skin-deep cosmetics don't change the true makeup of the underlying subject.

Screen scraping fortunately hasn't taken root in serious application development, instead the Web browser has become the modern terminal; HTML, the display format; and users have been delivered the graphical interface they yearned for. While true client/server developers were wrestling, solving issues such as systems management and distribution in a heterogeneous world, the Web filled in the software space created by the growth in the PC market and the availability of faster and cheaper networks.

Web applications still suffer from some of the problems that any page-based server GUI has, including the transactional nature of the screens, round-tripping for validation not delivering a highly responsible application, and the difficulty of working with multiple windows simultaneously. Most of the problems that plagued client/server development have been solved and several customers I talked to are reassessing the desktop because their applications have reached the physical boundaries of what a traditional J2EE program can deliver. There are more tools available now such as Java Web Start, better Swing libraries, and the Eclipse Rich Client Platform.

History has a habit of repeating itself, and in the past few months I've been alarmed to see demos of and read about several new ways to create a "rich desktop application" from within J2EE. All of these eloquently outline the current problems and limitations of the Web programming model from a user standpoint and preach the advantages of maximizing the power of the desktop. What is disturbing is that a lot of these then present a solution that is no more than 21st century screen scraping. Some of these offer solutions in which the same presentation markup can be rendered in a browser as a portlet or else in a desktop engine as the same GUI, but with native controls.

The debated advantages to this are that the investment in existing technology is preserved, and the same program can be deployed into a browser or to the user's desktop with the flick of a switch. My fear regarding these kinds of programs is that on the desktop they won't look and feel and behave as a true client program should, and because they falsely use adjectives such as "rich" or "desktop" to describe them, they'll somehow dilute these terms and make it hard for users to distinguish a dressed-up Web application from a true client one.

One of the advantages of having the browser shell is that every thing that lies within it is expected to operate in a certain way, and requests to the user to "press retry to refresh the page" or notifications that a "session time-out occurred" have become acceptable. Disguising the browser with native controls but not offering any of the advantages of a properly constructed client program offers a thin veneer that is no different from the screen scraping techniques of yore.

When the same server program can kick out a browser and rich GUI, is this an elegant solution that is going to meet the user's requirements, or is it just something that is technically elegant and demos well, but behind the robes is no more than lipstick on a browser?

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

joewinchester@sys-con.com

eclipseCON™ 2005

2nd Annual Gathering of the Eclipse Community

Plug-in to Eclipse

This is your opportunity to get in-depth technical information from the Eclipse experts, learn the latest tips and techniques for using the tools, network with fellow enthusiasts and experience the breadth and depth of the Eclipse community. Attending EclipseCon™ will expand your knowledge and make Eclipse work better for you.

Tim O'Reilly

Urs Hoelzle

John Wiegand

Erich Gamma

Keynotes: Tim O'Reilly,
CEO of O'Reilly Media

Urs Hoelzle,
VP of Engineering, Google

John Wiegand and Erich Gamma,
Eclipse Platform PMC Leaders

Over 60 Sessions and Tutorials for beginner through advanced participants. Some of the conference sessions and tutorials include: Eclipse Platform · Rich Client Platform (RCP) · Test and Performance Tools · Debugging · J2EE Development · Web Development and Tools · Building Plug-ins · Modeling Framework (EMF) · C/C++ Development · Business Intelligence and Reporting · Embedded Development

February 28-March 3, 2005
Burlingame, CA

Register at www.eclipsecon.org

A Time Zone Patch

A flexible solution

by Paulo Caroli

The java.util.TimeZone abstract class that represents a time zone is used to produce local time for a particular global time zone. A TimeZone comprises three basic pieces of information: an ID, a time zone offset, and the logic necessary to deal with DST (Daylight Savings Time).

While working with the TimeZone class provided with JRE 1.3 and 1.4, however, I discovered that it provides erroneous times for several time zones. This article describes this DST transition date problem, presents TimeZone test scenarios, and introduces TimeZonePatch. This patch is a flexible solution that enables the java.util.TimeZone to produce correct time zone data.

TimeZone Problem

I encountered the problem while working in a Java application that delivers Short Message Service (SMS) to wireless devices. The company's SMS application was being used in several countries and users were registering from several distinct time zones. A key requirement for the application is to have the user's local time as part of the SMS message.

The TimeZone JRE class was being used to retrieve the user's local time in the SMS message according to the user's registered time zone. It was discovered, however, that the application was delivering erroneous local times, and further investigation showed that the DST transition dates were not accurate for a number of time zones.

Testing TimeZone

The following test case was used to identify the time zone DST transition dates problem. It's based on the JUnit framework and is composed by TestTimeZone and DST classes (see Listings 1 and 2, respectively). The source code can be downloaded from www.caroli.org.

The correct DST entry and exit dates can be selected from trusted sources. For this article the dates were selected from the world clock Web site: www.timeanddate.com/worldclock/full.html.

To verify the DST transition problem the TestTimeZone test case was executed for two distinct time zones: America/Sao_Paulo and America/Santiago. The time zone data in the DST classes was retrieved from the world clock Web site.The tests results are depicted in Figures 1 and 2. Note that these tests are based on the JDK TimeZone class usage.

TimeZone Test 1:

America/Santiago time zone

```
public class DST{
  public static final String DATE_FORMAT =
"MM/dd/yyyy";
  public static final String START_DST =
"10/10/2004";
  public static final String END_DST =
"03/14/2004";
  public static final String TIMEZONE_
LOCATION = "America/Santiago";
}
```

TimeZone Test 2:

America/Sao_Paulo time zone

```
public class DST{
  public static final  String DATE_FORMAT =
"MM/dd/yyyy";
  public static final  String START_DST =
"11/02/2004";
  public static final  String END_DST   =
"02/15/2005";
  public static final  String TIMEZONE_
LOCATION  = "America/Sao_Paulo";
}
```

Test 2 for TimeZone shows the inaccuracy in the DST entry date of the TimeZone class with the America/Sao_Paulo time zone.

Patching TimeZone

The tests were executed with JRE 1.3 and 1.4 and both showed the same problem. Both versions of the TimeZone class read time zone IDs from the the tzmappings file in the jre/lib subdirectory. JRE 1.4 also contains time zone data files for various world regions in the jre/lib/zi subdirectory.

One solution for the 1.4 TimeZone class is to update the time zone files; however, these are binary and its change is dependent on you having access to and intimate knowledge of the JRE. The JRE provider could supply updates to the time zone files, however, it's not straightforward even for them to keep track of all the time zones' correct data. Although there are conventions about the DST transition dates and times, countries have the freedom to decide their own transition dates and times, and this has occurred in several places.

For these reasons I developed a patch for the SDK TimeZone, which solves the wrong DST transition dates problem independent of the original TimeZone implementation.

TimeZone Test Set-Up

```
public class TestTimeZone extends TestCase {
.
    protected void setUp() throws Exception
{
//        timeZone = TimeZonePatch.
getTimeZone(DST.TIMEZONE_LOCATION);
        timeZone = TimeZone.getTimeZone(DST.
TIMEZONE_LOCATION);
        super.setUp();
    }
.
}
```

TimeZonePatch

The TimeZonePatch class has been created for patching time zones according to an elected time zone data source. Listing 3 shows the TimeZonePatch code.

The fix works by having the TimeZonePatch class return your desired version of TimeZone. In case there is a time zone correction, this correction will be set in the timezonePatchList, which holds the patched TimeZone's objects. The TimeZonePatchListFactory class builds timezonePatchList.

A Factory of Your Time Zone Data

TimeZonePatchListFactory can get the time zone data from different sources. For example, the time zone data source can be an XML file, text file, database, Web service, or any other data source appropriate for your application.

Listing 4 shows TimeZonePatchListFactory, a Factory Method implementation that creates a time zone patch list based on an XML time zone data source. (Listings 4–6 can be downloaded from www.sys-con.com/java/sourcec.cfm.)

Listing 5 has timezonePatch.xml, an XML data source read by TimeZonePatchListFactory for patching the America/Sao_Paulo time zone.

Paulo Caroli, www.caroli.org, is the J2EE architect at blah!, a Telecom Italia Mobile (TIM) value-added services provider. Paulo has a master's degree in software engineering and is a Sun Certified Architect for Java Technology, with expertise in high-performance enterprise application development.

pcaroli@corp.blah.com

The presented TimeZonePatchListFactory implementation uses JOX (Java Objects in XML) library for reading the XML file into the TimeZonePatchList JavaBean.

When reading the XML document, a mapping between the root node name – TimezonePatchList – and the nested nodes – timezonePatch, dst, startDate, and endDate – is automatically made for the TimeZonePatchList bean, and the nested beans – TimeZonePatch, DST, and Date – respectively.

Testing TimeZonePatch

TimeZonePatch has also been tested by the same TestTimeZone test scenarios.

TimeZonePatch Test Set-Up

```
public class TestTimeZone extends TestCase {
.
    protected void setUp() throws Exception {
        timeZone = TimeZonePatch.getTimeZone(DST.TIMEZONE_LOCATION);
//        timeZone = TimeZone.getTimeZone(DST.TIMEZONE_LOCATION);
        super.setUp();
    }
.
}
```

The two distinct time zones, America/Sao_Paulo and America/Santiago, were retested, but now with the test class using TimeZonePath instead of the SDK TimeZone.

TimeZonePatch Test 1:

America/Santiago time zone

```
public class DST{
  public static final String DATE_FORMAT = "MM/dd/yyyy";
  public static final String START_DST = "10/10/2004";
  public static final String END_DST = "03/14/2004";
  public static final String TIMEZONE_LOCATION = "America/Santiago";
}
```

TimeZonePatch Test 2:

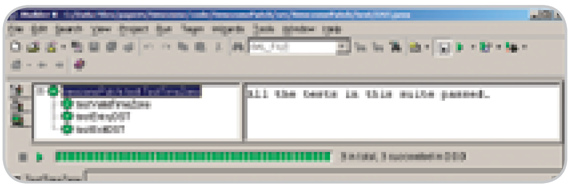
America/Sao_Paulo time zone

```
public class DST{
  public static final  String DATE_FORMAT = "MM/dd/yyyy";
  public static final  String START_DST = "10/17/2004";
  public static final  String END_DST  = "02/15/2005";
  public static final  String TIMEZONE_LOCATION  = "America/Sao_Paulo";
}
```

As depicted in TimeZonePatch test 1 and 2 (see Figures 3 and 4), the presented TimeZonePatch solution works properly for both time zones, proving to fix (according to the provided data source) the incorrect DST transition dates occurring in the SDK TimeZone class for the America/Sao_Paulo time zone (the identified incorrect TimeZone).

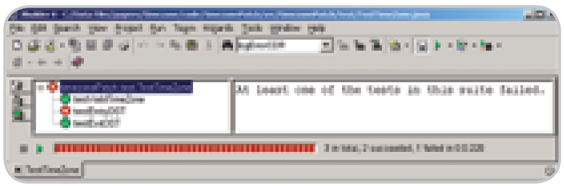
Conclusion

The TimeZonePatch presented in this article provides a flexible solution for patching the java.util TimeZone against incorrect time zone data. A factory of TimeZone patches, TimeZonePatchList-Factory, provides the ability to read correct time zone data from appropriate data sources or even create new time zones. Furthermore the presented solution patches only the time zones created by the factory, although it still works for the remaining SDK time zones.



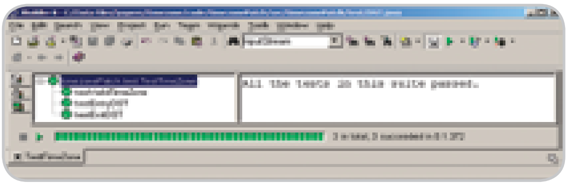
```
All the tests in this suite passed.
```

Figure 1 TimeZone Test 1 (America/Santiago DST transition) result



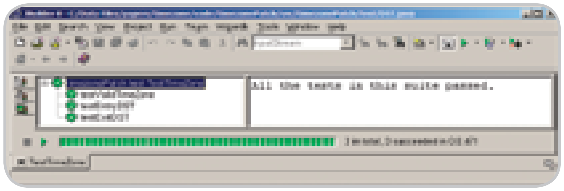
```
At least one of the tests in this suite failed.
```

Figure 2 TimeZone Test 2 (America_/Sao_Paulo DST transition) result



```
All the tests in this suite passed.
```

Figure 3 TimeZonePatch Test 1 (America/Santiago DST transition) result



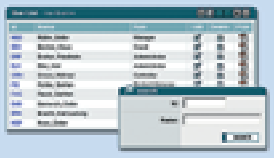
```
All the tests in this suite passed.
```

Figure 4 TimeZonePatch Test 2 (America/Sao_Paulo DST transition) result

COMMON CONTROLS www.common-controls.com

The Java™ Presentation framework for J2EE™ Web applications

Based on:
- Java™
- Servlets™
- Java Serverpages™
- and Struts



For Free!

Get your free trial version – www.common-controls.com
See the common controls in action – go for the Online Demo!

Contains the most common control elements which are required for the development of J2EE™ applications with rich HTML frontends like:

Lists | Trees | Tabfolders | Menu | Forms | BreadCrumbs | Calendar | Colorpicker

www.common-controls.com

References

- *JUnit framework:* www.junit.org
- *The source code for this article:* www.geocities.com/paulocaroli/src/timezonePatchCode.zip
- *The world clock Web site:* www.timeanddate.com/worldclock/full.html
- Gamma, E.; Helm, R.; Johnson, R.E.; and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.
- *JOX (Java Objects in XML):* www.wutka.com/jox.html
- *JOX download:* www.wutka.com/joxdownload.html
- *DTDParser download:* www.wutka.com/dtdparserdownload.html
- *Xerces-J download:* http://xml.apache.org/dist/xerces-j/Xerces-J-bin.1.4.4.zip
- Caroli, P. "Java Tip 138: Still parsing to generate your JavaBeans' XML representation?" JavaWorld, 2003: www.javaworld.com/javatips/jw-javatip138_p.html

Listing 1: TestTimeZone.java

```
package timezonePatch.test;

import junit.framework.*;
import java.util.Date;
import java.util.Calendar;
import java.util.TimeZone;

import timezonePatch.TimeZonePatch;

import java.text.SimpleDateFormat;

public class TestTimeZone extends TestCase {
    private TimeZone timeZone;
    public TestTimeZone(String name) {
        super(name);
    }

    protected void setUp() throws Exception {
//          timeZone = TimeZonePatch.getTimeZone(DST.TIMEZONE_
LOCATION);
        timeZone = TimeZone.getTimeZone(DST.TIMEZONE_LOCATION);
        super.setUp();
    }

    private boolean entryDST(TimeZone timezone, String entryDate,
String dateFormat) throws Exception {

        Date startDST = this.stringToDate(entryDate,dateFormat);
        //one day after DST
        boolean oneDayAfterExitDSTDate = timezone.
inDaylightTime(this.dayAfter(startDST));
        // one day before DST
        boolean oneDayBeforeEntryDSTDate = timezone.
inDaylightTime(this.previousDay(startDST));
        // correct values:
        // oneDayBeforeEntryDSTDate == false &&
        // oneDayAfterExitDSTDate == true
        return (!(oneDayBeforeEntryDSTDate) && (oneDayAfterExitD-
STDate));
    }

    private boolean exitDST(TimeZone timezone, String exitDate,
String dateFormat) throws Exception {
        Date endDST = this.stringToDate(exitDate,dateFormat);
        //one day after DST
        boolean oneDayAfterExitDSTDate = timezone.
inDaylightTime(this.dayAfter(endDST));
        // one day before DST
        boolean oneDayBeforeExitDSTDate = timezone.
inDaylightTime(this.previousDay(endDST));
        // correct values:
        // oneDayBeforeExitDSTDate == true &&
        // oneDayAfterExitDSTDate == false
        return ((oneDayBeforeExitDSTDate) && !(oneDayAfterExitD-
STDate));
    }

    public void testValidTimeZone() throws Exception {
        boolean expectedReturn = true;

        String[] allTimeZoneIds = timeZone.getAvailableIDs();
        boolean foundTimezone = false;
        for (int i=0; i<allTimeZoneIds.length; i++)
        {
            if(DST.TIMEZONE_LOCATION.equals(allTimeZoneIds[i]))
            {
              foundTimezone = true;
              break;
            }
        }
        assertEquals("valid timezone location ", expectedReturn,
foundTimezone);
    }

    public void testEntryDST() throws Exception {
        boolean expectedReturn = true;
        boolean actualReturn = this.entryDST(timeZone,DST.START_
DST,DST.DATE_FORMAT);
        assertEquals("DST enter date", expectedReturn, actualRe-
turn);
    }

    public void testExitDST() throws Exception {
        boolean expectedReturn = true;
        boolean actualReturn = this.exitDST(timeZone,DST.END_
DST,DST.DATE_FORMAT);
        assertEquals("DST exit date", expectedReturn, actualRe-
turn);
    }

    private Date stringToDate(String dateStr, String dateFormat)
throws
        java.text.ParseException {
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat(
dateFormat);
        Date dstChangeTime = simpleDateFormat.parse(dateStr);
        Date dstChangeTime_12HoursLater = new Date(dstChangeTime.
getTime()+1000 * 60 * 60 * 12);
        return dstChangeTime_12HoursLater;
    }

    private Date previousDay(Date date) {
        // Get yesterday's date.
        long time = date.getTime();
        return new Date(time - 1000 * 60 * 60 * 24);
    }

    private Date dayAfter(Date date) {
        // Get yesterday's date.
        long time = date.getTime();
        return new Date(time + 1000 * 60 * 60 * 24);
    }
}
```

Listing 2: DST.java

```
package timezonePatch.test;

public class DST{
  public static final String DATE_FORMAT = "MM/dd/yyyy";
  public static final String START_DST = "10/10/2004";
  public static final String END_DST = "03/14/2004";
  public static final String TIMEZONE_LOCATION = "America/
Santiago";
}
```

Listing 3: TimeZonePatch.java

```
 package timezonePatch;

import java.util.TimeZone;
import timezonePatch.model.*;
import timezonePatch.factory.*;

public abstract class TimeZonePatch extends TimeZone {
  private static TimezonePatchList timezonePatchList = null;
  public static TimeZone getTimeZone(String timeZone) {
    try{
      if (timezonePatchList == null) {
        timezonePatchList = TimeZonePatchListFactory.getTimezone-
PatchList();
      }
      if (timezonePatchList.exist(timeZone)) {
        return (TimeZone) timezonePatchList.getTimezonePatch(time
Zone);
      }
      // in  case timezone id is not in timezonePatchList,
      // the java.util.TimeZone is returned
      return TimeZone.getTimeZone(timeZone);
    }catch (TimeZonePatchListFactoryException e){
      e.printStackTrace();
      // in case TimeZonePatchList could not be sucessfully built,
      // the java.util.TimeZone is returned
      return TimeZone.getTimeZone(timeZone);
    }
  }
}
```

CTIA WIRELESS 2005®

A Division of CTIA–The Wireless Association™

The Most Important Technology Event of the Year!

March 14-16, 2005 Ernest N. Morial Convention Center New Orleans, LA **www.ctia.org**

REGISTER BEFORE JANUARY 31, 2005 and Save up to $345!

CTIA's 12th Annual Wireless Gala

The wireless industry's premiere networking event is always the hottest ticket in town!

For sponsorship information contact Bonnie Knight at bknight@ctia.org.

Keynotes

Stay tuned for more exciting Keynote announcements...

Daniel A. Carp
Chairman and Chief Executive Officer
Eastman Kodak Company

Jeong Han Kim
President
Samsung Telecommunications America

Steve Largent
President/CEO
CTIA-The Wireless Association

CTIA WIRELESS 2005

is the most important global event where standards bodies converge to set the agenda for the future of the wireless industry. Network with the largest gathering of wireless engineers and technologists to find the tools you need to help advance the wireless industry.

CTIA WIRELESS 2005 offers numerous opportunities for the wireless engineer and technologist:

- ➡ **CTIA Education** – *a track dedicated to wireless technology*
- ➡ **Tower Summit & Trade Show 2005** – *premiere wireless infrastructure event*
- ➡ **IEEE Wireless Communications Network Conference (WCNC) 2005** – *the industry's foremost conference for developing wireless standards and engineering*
- ➡ **Developers Conferences**
- ➡ **Exhibits** – *a 400,000 sq. ft. floor displaying the latest in wireless technology*
- ➡ **Wireless News** – *More new wireless product announcements than any other show!*
- ➡ **CTIA Wireless Home** – *a 7,000 sq. ft. home that brings wireless technology to life*

Produced by
CTIA
The Wireless Association™

JBuilder 2005 by Borland Software Corporation

Reviewed by
Ken Sipe

There is an old joke in which a man is seen coming from a conservation area with a hefty catch of fish. After noticing the proficiency of the fisherman when others had come back empty-handed, a park ranger decides to tag along with him. When they get to a specific location, the fisherman lights a stick of dynamite and throws it in the water, retrieving a huge quantity of fish. While the ranger begins to explain the illegal nature of the fisherman's actions, the man lights another stick, hands it to the ranger, and says, "Do you want to talk?…or do you want to fish?" Borland, with JBuilder, seems to ask a similar question, "Do you want to play, arrange, and mess with your development environment?…or do you want to code?"

It isn't possible to provide an in-depth review of all of JBuilder's features in this short article. While preparing to write this review, I downloaded JBuilder 2005's feature matrix and found it was 32 pages! This article reviews JBuilder 2005 Enterprise Edition, which was released to the public August 2004. However, since I was in the group of beta testers, I had used JBuilder 2005 for several months prior to its public release.

Editor Features

Significant improvements can be found in the editor. MemberInsight has been updated to become Smart MemberInsight. Instead of the drop-down displaying all methods for a given object, it's filtered to the most likely candidates. This can be very confusing at first and takes some getting use to. Depressing ctl+h will expand the filtered list to all possibilities. Key mappings can be adjusted to provide the previous behavior.

ErrorInsight is fairly new in JBuilder and provides a welcomed approach to agile development. Essentially, any errors written in code will be highlighted by ErrorInsight flags. Alt+enter invokes the ErrorInsight to provide options; for example, when used with a TDD approach, referencing classes that don't exist yet causes JBuilder to ask if it should search for the reference or create a new class. Similarly, if a method is used that throws an exception, the provided options (depending on code location) are "Add Throws Clause," "Add Catch Clause," and "Surround with Try/Catch" (see Figure 1).

Development help can also be gleaned from code audit. This new feature highlights potential concerns. For instance, a comparison of two strings with the "==" operator is usually a mistake. With code audits, this code is highlighted with a description of the issue. This feature can get annoying depending on code style. For instance, it complains that code after an if statement needs to be in a code body. Each of these audits can be configured on a project-by-project basis, but again illustrates another deficiency: there doesn't appear to be a way to export and import preferred audit lists for a department so it can be used in another project. Hopefully this will be extended in the future to work like ErrorInsight. If the tool knows that code audit's preference is to put code in a code block after an if statement, why doesn't it assist with that? This is still a welcome feature though as many projects spend thousands of dollars on products for this specific item.

It should be mentioned that some of the best features of JBuilder are not intuitively obvious, like intelligent code editing. For instance, the smart paste feature, through refactoring, cuts several lines of code and pastes them in another class, and JBuilder asks if the associated imports should be added.

JDK 5

A noteworthy must-have for JBuilder 2005 is its support for JDK 5. Provided support includes help iden-

Borland Software Corporation

100 Enterprise Way
Scotts Valley, CA 95066-3249
Phone: 800 632-2864
E-mail: customer_service@borland.com
Web: www.borland.com

Specifications Platforms

Windows 2000 – Windows XP, OSX, Solaris 9, Linux (official support for Sun Java Desktop and Red Hat Enterprise Linux 3)

Pricing

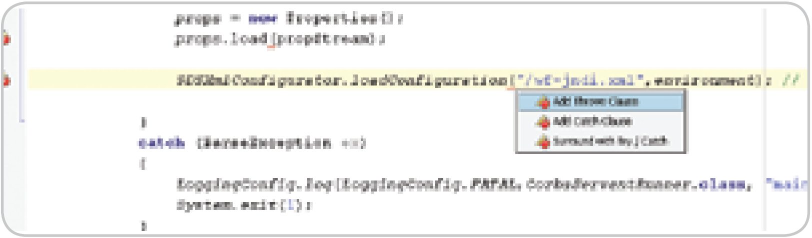
Enterprise: new – $3,500; upgrade – $1,900.
Developer: new – $500; upgrade – $299.
Foundation: free (including for commercial use)

Test Environment

Dell Inspiron 8200, 2GHz Intel P4 processor, 60GB disk, 1GB of RAM, Windows XP

Ken Sipe is the CEO and founder of Code Mentor, Inc. He provides coding, mentoring, and training in OOAD and Java technologies. In addition, Ken is an internationally recognized speaker in the field of distributed computing, speaking often at JavaOne, NFJS, and BorCon.

kensipe@codementor.net



```
        props = new Properties();
        props.load(propStream);

        [illegible]Configurator.loadConfiguration("/wf-jndi.xml",environment); //

    }
    catch (ParseException ex)
    {
        LoggingConfig.log(LoggingConfig.FATAL, CorbaServantRunner.class, "mai
        System.exit(1);
    }
```

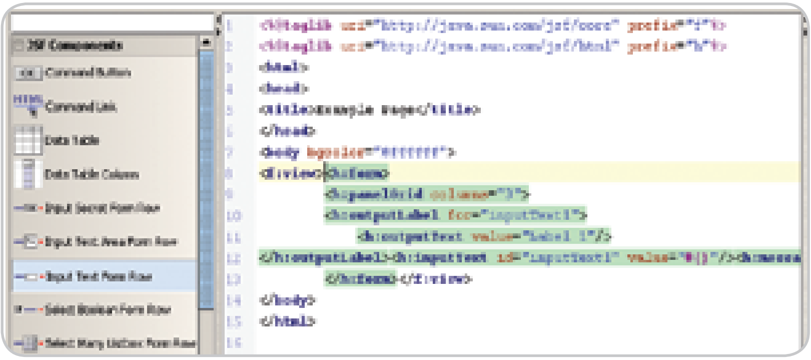
Figure 1 ErrorInsight example

JAVA DEVELOPER'S JOURNAL Advertiser **Index**

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	4
Common Controls	www.common-controls.com	+49 (0) 6151/13 6 31-0	45
CTIA Wireless 2005	www.ctia.org	301-694-5243	47
EclipseCon 2005	www.eclipsecon.org	503-252-2458	43
EV1 Servers	www.ev1servers.net	800-504-SURF	13
Fitech Laboratories, Inc.	www.fitechlabs.com/grid	646-495-5076	27
Google	www.google.com/jdj	650-623-4000	33
Information Storage & Security Journal	www.issjournal.com	888-303-5282	49
InstallShield	www.installshield.com/jdj	800-809-5659	21
InterSystems	www.intersystems.com/match3	617-621-0600	Cover IV
IT Solutions Guide	www.sys-con.com/itsolutions	888-303-5282	61
LinuxWorld Conference & Expo	www.linuxworldexpo.com	508-424-4810	41
M7	www.m7.com/d7.do	866-770-9770	29
Microsoft	msdn.microsoft.com/visual		Cover II
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	39
Parasoft Corporation	www.parasoft.com/jtest_jdj	888-305-0041	9
PointBase	www.pointbase.com	877-238-8798	37
ReportingEngines	www.reportingengines.com/download/f1ere.jsp	888-884-8665	17
Software FX	www.softwarefx.com	800-392-4278	Cover III
WebRenderer	www.webrenderer.com		7
Web Services Edge 2005 East	www.sys-con.com/edge	201-802-3069	51-60

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of ***Java Developer's Journal.*** All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in ***Java Developer's Journal.*** Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.

This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.



```
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<html>
<head>
<title>Example Page</title>
</head>
<body bgcolor="#ffffff">
<f:view><h:form>
        <h:panelGrid columns="3">
        <h:outputLabel for="inputText1">
            <h:outputText value="Label 1"/>
</h:outputLabel><h:inputText id="inputText1" value="#{}"/><h:messa
        </h:form></f:view>
</body>
</html>
```

Figure 2 JSF Smart Tags

tifying conflict issues for situations in which you name a variable enum (which is now a keyword). It also provides support for generics, foreach, autoboxing, and the added Javadoc tags. It even provides a refactoring wizard to migrate iterating loops to the new foreach approach. However, its JDK 5 support isn't complete: among the deficiencies is the lack of support for static imports.

Web Development

The largest collection of must-have features for JBuilder 2005 is that which includes the Web development tools. They provide support for JSP 2.0 and Servlet 2.4 through support for Tomcat 5.0, including JSR 45 debugging support for debugging JSP pages, JavaServer Faces, and Web services enhancements.

JavaServer Faces support was initially a disappointment. However, during Borland's conference the developer of the tool demonstrated some otherwise hidden features in JBuilder, which raised the level of value that JBuilder provides for JSF development. While most other editors are providing design-time WYSIWYG GUI tools for JSF, JBuilder doesn't. The upside to this approach isn't self-evident. The JSF specification doesn't standardize tool support, so whatever tool you use, it has its own proprietary approach for designing JSF. The pro with JBuilder is it doesn't lock you in and you can use JBuilder to work on projects originating in another editor. The downside is WYSIWYG is nice and isn't available here. Of course, maybe we should be upset with the JCP on this one. JBuilder's JSF support is geared to work specifically with the reference implementation from Sun. I've had some success in getting it to work with MyFaces implementation, but you give up some features when doing so.

The tag component palette initially seems like a bad idea, at least for seasoned developers, as it takes up valuable screen real estate and provides little value…until you drag a "smart" component onto a JSP page (I'm still not sure how to distinguish the smart tags from the dumb tags). Try adding a JSF Input Text Form Row to a JSP page; it knows that it's required to be in a form and that form must be in a view. The text in the editor will dynamically change depending on its placement to ensure the nesting of tags is correct. Looking at Figure 2, the inserted tag knows it needs more tags in order to work. It also knows that it already is in the view tag, which would also be inserted if it weren't present. Very cool. With the added benefit of Tag Insight (which essentially is ErrorInsight for tags), the Web development benefits for 2005 make it worth the upgrade.

Additional Features

One of my favorite new features doesn't seem to be documented in what's new. JBuilder now refactors references in XML to classes, which are refactored. This means that struts-config.xml, web.xml, and all the EJB XML files are correctly updated when a class is renamed or repackaged. This is a huge time-saver.

Other new features worth the upgrade: Web services WS-I testing tools, distributed refactoring, debugging Ant and Ant tasks, and Subversion support.

What to Do...What to Do

The Foundation Edition is a free download from Borland and includes many of the editor features as well as most of the refactoring tools. Notably missing is any Web development or EJB development tools. This edition is a good choice for beginners, organizations that aren't doing Web or enterprise development, or those environments that are using other tools for enterprise development such as xDoclet.

The Developer Edition packs the punch in this release. If you're not doing EJB, CORBA, or Web services development, most of the best features exist in this version.

Of course, if you're like me and you don't know what your next server will be, you have to maintain multiple servers and you don't want to waste time tinkering with configurations, or you need one or more the following – Web services tools, advanced OptimizeIT tools (Thread debugger), advanced XML tools, or advanced JUnit tests – you'll need to get the Enterprise Edition. This is the best option for less tinkering and more coding.

JDJ Product Snapshot

Target Audience: Java programmers
Level: Beginner to advanced
Pros:
- Best choice for consultants and shops required to support multiple application server environments
- JDK 5 support
- JSF support
- Project structure flexibility
- Refactoring

Cons:
- No support for application servers and frameworks released after JBuilder's release, which includes Tomcat 5.5, WebSphere 6.0, and JBoss 4.0

See what everyone is talking about… ***The Application Server Shoot-Out!***

International Web Services Conference & Expo

web services EDGE conference&expo

Register by JANUARY 21, 2005 SAVE Up To $300

WEB SERVICES EDGE 2005 EAST

Keynotes

Matt Ackley
eBay

Ari Bixhorn
Microsoft

Mike Milinkovich
Eclipse.org

Product Teams Delivering SOA, Web Services, Java, .NET, & XML Solutions

New for 2005…

- Cabana Night Developer Exchange
- Application Server Shootout
  facilitated by Burton Group
- Hands On VS and .NET Labs
- Software Giveaway

Free Tutorials presented by critical sites Novell Microsoft Java Community Process

For more information:
visit: www.sys-con.com/edge
call: 201-802-3066
e-mail: jimh@sys-con.com

February 15-17, 2005
John B. Hynes Convention Center
Boston, MA

PRODUCED BY: SYS-CON MEDIA
PRESENTED BY: SYS-CON EVENTS

DIAMOND SPONSOR: Microsoft
GOLD SPONSORS: Novell SAP
ASSOCIATION SPONSORS: INTEGRATION CONSORTIUM OASIS ineta eclipse
CORPORATE SPONSOR: Java Community Process
APP.-SERVER SHOOTOUT FACILITATOR: burton group
MEDIA SPONSORS: Information Storage + Security Journal, JDJ, WebServices Journal, .NET Developer's Journal, queue, CoDe, XML-Journal, WLDJ, asp.netPRO, WebSphere Journal, ColdFusion Developer's Journal, SD Times, MX developer's journal

SPECIAL INSERT: WEB SERVICES EDGE 2005
WWW.SYS-CON.COM/EDGE
FEBRUARY 15-17, 2005
JOHN B. HYNES CONVENTION CENTER • BOSTON, MA

SPECIAL INSERT: WEB SERVICES EDGE 2005
WWW.SYS-CON.COM/EDGE
FEBRUARY 15-17, 2005
JOHN B. HYNES CONVENTION CENTER • BOSTON, MA

International Web Servi

web services EDGE conference&expo

Web Services Edge 2005 East

February 15–17, 2005
John B. Hynes Convention Center
Boston, MA

The Largest *i*-Technology Event of the Year!

Who Should Attend

- Software Developers
- Software Engineers
- Application Developers
- Development Managers
- Technical Directors
- Analysts/Programmers
- IT Managers
- CEOs
- CTOs
- CIOs
- Technical Architects
- Team Leaders
- Software Consultants
- IT Directors
- Project Managers

Features & Attractions

- 3 Days Packed with Education and Training
- Keynotes & Panel Discussions from Industry Leaders
- 50+ Hard-hitting and Informative Seminars
- FREE Web Services Tutorial Presented by Novell
- FREE .NET Tutorial Presented by Microsoft
- FREE Web Services Security Tutorial Presented by Critical Sites
- Cabana Night Developer Exchange Presented by INETA
- Panel Discussions Presented by JCP
- Opening Night Welcome Reception
- Compelling Case Studies & Best Practices
- Hands-On Labs Featuring .NET and Visual Studio Presented by Microsoft Partners
- Featured Product Demonstrations
- Exhibit Floor featuring hundreds of products
- Real-time SYS-CON Radio Interviews

VISIT **www.sys-con.com/edge** FOR TIMES & SCHEDULES

The program, including topics and times, is subject to change. Please refer to www.sys-con.com for all updates.

ces Conference & Expo

Keynote Speakers

Tuesday, February 15, 11 a.m.

Matt Ackley
Senior Director, eBay Developers Program

Web Services for eCommerce

eBay, The World's Online Marketplace, has more than 114 million registered users, 10,000 developers, and over 700 live, third-party applications. Four years ago, eBay began allowing third parties to build applications that tap into eBay, and today eBay hosts one of the leading Web services platforms. Through its developer program, eBay enables third parties to create cutting-edge Web services applications that benefit the buyers and sellers on eBay. At present, 40% of eBay's listings come through its API, which handles more than a billion Web services calls a month. Ackley will discuss the rewards and challenges of building and maintaining one of the world's leading Web services platforms, and share insights and practical guidelines for others.

Matt Ackley is senior director of the eBay Developers Program. He supports eBay's vision to be the leading platform for global online commerce, and is chartered with creating a thriving ecosystem between eBay, its community of users, and third-party developers and solution providers. Ackley joined eBay in 2003 as part of eBay's acquisition of FairMarket, which provided technology solutions and services to online marketplaces.

Wednesday, February 16, 11 a.m.

Ari Bixhorn
Director, Web Services Strategies, Microsoft Corporation

Introducing Indigo: The Unified Programming Model for Building Service-Oriented Applications

Indigo is Microsoft's unified programming model for building service-oriented applications on the Windows platform. It enables developers to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing investments. Indigo combines and extends the capabilities of existing distributed application technologies, including .NET Enterprise Services, System.Messaging, Remoting, ASMX, and WSE to deliver a unified development experience spanning distance, topologies, hosting models, protocols, and security models. This keynote will provide an inside look at Indigo and show you how Indigo will radically simplify the development of distributed, service-oriented applications.

Ari Bixhorn is the director of Web Services Strategy in the Developer and Platform Division at Microsoft Corp. He is responsible for product planning and technical evangelism for Microsoft's Web services offerings, including "Indigo," the code name for a component of the next version of the Windows operating system, code-named Windows "Longhorn." Bixhorn has spent the past five years at Microsoft, driving product management efforts for the Visual Basic and Visual Studio development systems.

Thursday, February 17, 11 a.m.

Mike Milinkovich
Eclipse.org

An Open Development Platform for Web Services

Open source technology runs the Internet. Linux, Apache, PHP and Eclipse are highly successful open source communities that provide the backbone for today's Web applications. All indications point to a continued value proposition for organizations for leveraging open source when developing and deploying SOA-based applications. This keynote will examine the benefits of using open source technologies, the decision-making process used when adopting these solutions and the potential for contributing back to the open source community.

Mike Milinkovich has held key management positions at Oracle, WebGain, The Object People, and Object Technology International Inc. (which subsequently became a wholly owned subsidiary of IBM), assuming responsibility for development, product management, marketing, strategic planning, finance, and business development. Mike earned his MS degree in information and systems sciences and a bachelor of commerce degree from Carleton University in Ottawa, Canada.

VISIT www.sys-con.com/edge FOR TIMES & SCHEDULES

The program, including topics and times, is subject to change. Please refer to www.sys-con.com for all updates.

SPECIAL INSERT: WEB SERVICES EDGE 2005
WWW.SYS-CON.COM/EDGE
FEBRUARY 15-17, 2005
JOHN B. HYNES CONVENTION CENTER • BOSTON, MA

DAY 1 FEBRUARY 15

	Java	.NET		Web Services
7:30	Registration			
8:00	**FREE Tutorial** – Ashish Larivee, Novell, **Using a Web Services Framework to Build SOA Applications**			
9:00	(J-1) What's New In JDO 2.0	(.NET-1) Intro to SPOT		(WS-1) Ensuring Web Services Interoperability
10:00	(J-2) Using Java Messaging in Real-Time Trading Systems	(.NET-2) An Introduction to SQL Server Reporting Services		(WS-2) Web Services Standards: Going Behind the Mask
11:00	**Opening Keynote** – Matt Ackley, Senior Director, eBay Developer Program, eBay			
12:00	**EXPO OPEN** (12 P.M.–5 P.M.)			
3:00	**Keynote Panel** Presented by JCP – **Web Services and Security** Moderator: Onno Kluyt, Sr Director & Chair, JCP Program, Sun Microsystems			(WS-2B) Solving Complex Business Problems Though SOA
4:00	(J-3) The ROI of a Java-Rich Client	(.NET-3) Go With The Flow – Human Workflow Services in BizTalk 2004	(.NET-3B) Techniques with Visual Basic.NET	(WS-3) The XML Data Challenge
5:00	Opening Night Reception			

DAY 2 FEBRUARY 16

	Java	.NET		Web Services
7:30	Registration			
8:00	**FREE Tutorial –** Thom Robbins, Microsoft – **The Next Generation of Visual Studio** (free with VIP preregistration)			
9:00	(J-4) Web Services End-to-End Security on J2EE: Gaps and Proposed Solutions	(.NET-4) The Microsoft Framework: An Agile Software Development Process for Building Web Service Applications		(WS-4) How To Bulletproof Your Web Services
10:00	(J-5) J2ME and Eclipse	(.NET-5) Securing Service-Oriented Architecture with Microsoft's WSE 2.0		(WS-5) The Role of Policy in Web Services Integration – It's More Than Just Security
11:00	**Keynote** – Ari Bixhorn, Director, Web Services Stategies, Microsoft Corporation			
12:00	**EXPO OPEN** (12 P.M.–4 P.M.)			
3:00	**Application Server Shootout**			
4:00	(J-6) The Impact of JBoss and Mono on the Application Server Market	(.NET-6) Web Services Security for Dummies with WSE2	(.NET-6B) J2EE to .NET Interoperability and App. Integration	(WS-6) B2B Policy Enforcement: The Third Rail of SOA Implementation
5:00	(J-7) Migrating Enterprise Applications Between J2EE Application Servers	(.NET-7) So You THINK You Know What an Object Is…	(.NET-7B) Building and Using Advanced ASP.NET Web Controls	(WS-7) Driving SOA Governance
6:00	**Cabana Night –** Hosted by INETA			

DAY 3 FEBRUARY 17

	Java	.NET	Web Services
7:30	Registration		
8:00	**FREE Tutorial** – Patrick Hynds and Duane Laflotte, Critical Sites – **Security, The New Reality** (free with VIP preregistration)		
9:00	(J-8) Design Patterns and Project Organizational Techniques for "Write Once, Debug Everywhere"	(.NET-8) Migrating ASP to ASP.NET	(WS-8) SOA: From Pattern to Production
10:00	(J-9) Using Grid Computing with Web Services and J2EE to Create Internet-based SOAs	(.NET-9) Smart Client Development with the Offline Application Block	(WS-9) High Performance Web Services – Tackling Scalability and Speed
11:00	**Keynote** – Mike Milinkovich, Executive Director, Eclipse Foundation		
12:00	**EXPO OPEN** (12 P.M.–4 P.M.)		
3:00	(J-10) Java Web Services Programming Tips & Tricks	(.NET-10) CLR Internals	(WS-10) So You Want an SOA: Best Practices for Migrating Toward Service Orientation in the Enterprise
4:00	(J-11) JCP Program: How the Java Technology Binary Software Standard is Managed and Evolves	Visit Web site for update	(WS-11) Four Abilities SOA Will Lack Without a Registry

The program, including topics and times, is subject to change. Please refer to www.sys-con.com for all updates.

WS Security	Case Study
FREE Tutorial –Aaron Williams, JCP, **Developing Web Services Using Java Technology**	
(WSS-1) Identity in SOA	(CS-1) Developing E-Commerce Applications with Web Services
(WSS-2) Securing Web Services with WS-Security	(CS-2) Developing Enterprise Class Web Services
	(CS-3) Service-Oriented Development on NetKernel – Patterns, processes and product to reduce the complexity of IT systems
(WSS-3) Anatomy of a Web Services Attack	

Novell

FREE Web Services Security Tutorial Presented by Novell

Using a Web Services Framework to Build SOA Applications

Tuesday, Feb. 15
8:00 A.M. – 11:00 A.M.

FREE Tutorial Presented by Java Community Process

Developing Web Services Using Java Technologies

Tuesday, Feb. 15
8:00 A.M. – 11:00 A.M.

Java Community Process

Free Tutorials with VIP Preregistration ONLY!

WS Security	Case Study
(WSS-4) Using Mobile Phones as an SSO Authentication Device in SOA Solutions	(CS-4) Orchestrating FORCEnet Engagement Packs with BPEL for Web Services
(WSS-5) Building Intelligent Enterprises with Novell's Identity-Driven Computing	(CS-5) CPI: A Globally Integrated Problem-Tracking and Resolution System Using Java Web Services
(WSS-6) XML Content Attacks	(CS-6) The Transformation of SiteRefresh into a Web Services
(WSS-7) The Interoperability Challenge of Web Services Security Standards	

Microsoft

FREE
.NET Tutorial Presented by Microsoft

The Next Generation of Visual Studio
Thom Robbins

Wednesday, Feb. 16
8:00 A.M. – 11:00 A.M.

Free Tutorials with VIP Preregistration ONLY!

WS Security	Case Study
(WSS-8) Transitioning Successfully to SOA and Web Services: Building the Infrastructure for SOA Growth	(CS-8) Using SOA and Web Services to Issue Business Licenses in the District of Columbia
Visit Web site for update	(CS-9) Developing Web Services with Eclipse

critical sites™

FREE
.NET Tutorial Presented by Critical Sites

Security, The New Reality
Patrick Hynds

Thursday, Feb. 17
8:00 A.M. – 11:00 A.M.

Free Tutorials with VIP Preregistration ONLY!

The program, including topics and times, is subject to change. Please refer to www.sys-con.com for all updates.

JAVA TRACK

J-1 What's New In JDO 2.0

Patrick Linskey, SolarMetric

Tuesday, Febuary 15, 2005 9:00 A.M. – 9:50 A.M.

In this presentation, SolarMetric's CTO Patrick Linskey introduces JDO, the standards-based object persistence specification, and the new changes introduced with the release of JDO 2. Developers using JDO are seeing increased application portability, reduced development cycle time, improved code quality, higher performance, and more manageable scalability. The session will highlight the problems of persisting data to a database and the basics of JDO. Patrick will focus on what's new with JDO 2, including the improvements and alternatives to the JDO Query Language, detached database operations, and the new metadata for object / relational mapping.

BIO: Patrick Linskey has been working with Java Data Objects for over 3 years and has been involved in object/relational mapping for 5+ years. As the founder and CTO of SolarMetric, Patrick drives the technical direction of the company. Patrick is a luminary on JDOcentral, a consortium committed to marketing the JDO standard. He has been one of the leaders on the JDO specification team, currently helping to drive the JDO 2.0 specification.

J-2 Using Java Messaging in Real-Time Trading Systems

Yakov Fain, Smart Data Processing, Inc.

Tuesday, Febuary 15, 2005 10:00 A.M. – 10:50 A.M.

Any modern financial trading system is a complex distributed application that runs on multiple platforms and consists of components and services that have to communicate with each other. This presentation is about using JMS and Message-Oriented Middleware as a backbone that provides a fast and reliable delivery mechanism between various components and tiers of such systems. This presentation is based on my real-world experience in design, development, and production support of an application that had to wire together midrange computers with J2EE and LDAP servers, non-Java stock exchange software, and mainframe legacy reporting systems.

BIO: Yakov Fain works as a Java architect for a major bank in New York City. He is a member of the editorial board of *JDJ*. Yakov has 23 years of experience working in software development.

J-3 The ROI of a Java-Rich Client

Coach Wei, Nexaweb Technologies

Tuesday, Febuary 15, 2005 4:00 P.M. – 4:50 P.M.

Rich client, thick client, thin client – what are they? What are the benefits of rich client, in particular Java Rich Client? This session presents an overview of the various client technologies, in particular various Java-based rich-client solutions, including Swing/AWT, SWT, and XML. Further, it analyzes the trade-off between these different Java-rich client technologies, and presents real-world case studies to justify the ROI of Java-Rich Client solutions.

BIO: Coach Wei is founder and CTO of Nexaweb Technologies (www.nexaweb.com), which develops the leading software platform for building and deploying Enterprise Internet Applications. Coach has his master's degree from MIT, holds several patents, is the author of many technology publications, and is an industry advocate for the proliferation of open standards.

J-4 Web Services End-to-End Security on J2EE: Gaps and Proposed Solutions

Sudhir Bhojwani, Sushil Shukla, & Sudhrity Mondal, BEA

Wednesday, Febuary 16, 2005 9:00 A.M. – 9:50 A.M.

Even though WS-* security standards (WS-Security, WS-Trust, WS-SecureConversation, WS-Policy, etc.) are sufficiently prescriptive on specific security subjects like signing, SOAP message encryption, request/receive security tokens, they do not provide end-to-end security protocol that Web services can depend on to meet their security requirements. The most significant gap is identity propagation from a Web service into a J2EE container. Current JAX-RPC specification or JSR 109 does not cover this issue. This presentation identifies the gaps and discusses the approaches to plug these gaps. It also discusses implementation of a solution for identity propagation from client to Web service and from Web service to J2EE container.

BIO: Sudhir Bhojwani is a senior architect working for BEA Technical Solutions Group. He has over eight years of industry experience architecting and designing systems based on component technologies like CORBA and EJBs, and most recently working on SOA principles.

BIO: Sushil Shukla is a principal architect working for BEA Technical Solutions Group. He has 19 years of industry experience and 9 years of architecting and designing application server-based large systems. He has extensive experience with tools and technologies used in developing mission-critical applications.

BIO: Sudhrity Mondal is a principal architect working for BEA Technical Solutions Group. He has 14 years of industry experience and specializes in Java technology, enterprise integration solutions, object-oriented software design, and development and software patterns. He has helped prominent BEA Customers to architect, design, and implement new systems using Tuxedo, J2EE, and Web services over the past eight years.

J-5 J2ME and Eclipse

Michael Van Meekeren, IBM

Wednesday, Febuary 16, 2005 10:00 A.M. – 10:50 A.M.

Eclipse provides support for Java program development such as editing, compiling, and debugging, and is readily extensible through its plug-in mechanism. Many have been involved in the development of plug-ins that support the building and launching of embedded applications (with support for various platforms, such as J2ME/MIDP, PocketPC, and PalmOS). This talk will show how applications can be developed, compiled, analyzed, and compressed to fit on really small devices. It will include reports on practical experience, it will provide background information on developing Java applications for resource-constrained environments, and explain what Java standardization processes are under way.

BIO: Michael Van Meekeren has been a senior developer with IBM Ottawa Labs (formerly known as Object Technology International) since 1994, and has played an active role in the development of Envy/Developer, IBM Smalltalk, VisualAge for Java, and WebSphere Studio Device Developer. Michael is currently the IBM Eclipse Platform UI Team Lead at IBM Ottawa Labs.

J-6 The Impact of JBoss and Mono on the Application Server Market

Pierre Fricke, D. H. Brown Associates

Wednesday, Febuary 16, 2005 4:00 P.M. – 4:50 P.M.

Linux and Apache drove a dramatic change in the server operating system and Web server marketplaces. These areas, dominated by Microsoft and Sun in the late 1990s, now see leading open source alternatives challenging these leaders. But open source's impact doesn't stop there. JBoss, the open source J2EE platform, is becoming the high-volume leader in the J2EE application space. More recently, Mono, the open source implementation of Microsoft .NET, promises to be a main strategic item for Novell. Mono may extend Microsoft's hegemony into Linux and open-source by countering J2EE's cross-platform strategic advantage.

BIO: Pierre Fricke, D.H. Brown's vice president of Application and Integration Infrastructure, extends the company's unique technical and strategic analysis into the J2EE, Microsoft .NET, and integration infrastructure space. After completing his M.B.A., Pierre became one of the leading strategists and marketing leaders in IBM focusing on interoperability, integration, WebSphere, Windows NT, UNIX, as well as Linux and open source.

J-7 Migrating Enterprise Applications Between J2EE Application Servers

Ajit Sagar, Infosys Technologies, Ltd

Wednesday, Febuary 16, 2005 5:00 P.M. – 5:50 P.M.

This session will provide guidelines, best practices, and a methodology to tackle a problem that is sapping the budgets of enterprise which have invested heavily in J2EE technology – the migration of enterprise applications between J2EE application servers. The strategy and planning for such initiatives is very complex and requires planning in advance. Enterprise applications, once deployed, have a multitude of dependencies, besides the dependency on Java APIs. The drivers for migration can include version upgrades, corporate agenda, maintenance costs, industry alliances, rapid upgrades to the J2EE platform APIs, etc. A planned migration ensures a successful implementation while minimizing the impact. This session will describe a strategy to plan for the migration of large portfolios of applications between application server vendors, between application server versions, and between hardware platforms. Real world examples of how this strategy has been applied in the industry will be provided.

BIO: Ajit Sagar is a senior technical archiect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has been working with Java since 1997, and has more than 15 years' experience in the IT industry. Ajit has served as *JDJ*'s J2EE editor, was the founding editor of *XML-Journal*, and has been a frequent speaker at SYS-CON's Web Services Edge series of conferences. He has published more than 75 articles.

J-8 Design Patterns and Project Organizational Techniques for "Write Once, Debug Everywhere"

Dr. Michael Juntao Yuan, University of Texas

Thursday, Febuary 17, 2005 9:00 A.M. – 9:50 A.M.

Unlike the wildly successful server-side Java technology, the true "write once run anywhere" vision has never been achieved for client-side Java. For Java developers offering end-to-end smart client–based SOA solutions, the development process is still "write once, debug everywhere." As the client-side platforms evolve from a handful of PC OSs to hundreds of devices with different OSs and UI characteristics, developers must leverage design patterns and innovative project organizational techniques to smooth the development and optimization process. This session will introduce more than a dozen of those design patterns and organizational techniques based on Michael's experience working with Nokia developers to develop scalable mobile enterprise applications.

BIO: Dr. Michael Juntao Yuan is an author, developer, and software architect for end-to-end mobile software. He is a contributing editor to *JDJ* and a frequent contributor to many developer forums and publications. He is the author of two books.

J-9 Using Grid Computing with Web Services and J2EE to Create Internet-based SOAs

Kieran Taylor, Akamai

Thursday, Febuary 17, 2005 10:00 A.M. – 10:50 A.M.

Service-oriented architecture, Web services, and J2EE technologies are dramatically changing the ways in which enterprises develop and deploy their Internet-facing applications. Because these applications potentially have a global user base, correctly architecting applications is a particular challenge. A combination of grid computing and utility computing offers a way to provide computing resources when and where they are needed, but developers must factor in certain considerations during design. This session will provide an in-depth overview of three real-world case studies using Grid computing in combination with Web services and J2EE to create Internet-based SOAs. It is designed for application architects and developers, and attendees will learn how applications can be designed to operate in a distributed computing environment such that performance and scalability problems are bypassed during deployment.

BIO: Kieran Taylor is currently the director of product management for Akamai Technologies and focuses strategy and direction for Akamai's emerging technologies, including its J2EE Internet applications delivery services. While at Akamai, Taylor has helped promote the open-standard Edge Side Includes (ESI), a markup language for dynamic content assembly and delivery at the edge, in use today by many companies.

J-10 Java Web Services Programming Tips & Tricks

André Tost, IBM

Thursday, Febuary 17, 2005 3:00 P.M. – 3:50 P.M.

This session will present a collection of programming tips and tricks related to consuming and providing Web services in Java. This collection has been created by a number of developers and consultants and is the result of many real-life project experiences. We will focus on implementation aspects for Web services and not go into any detail on architecture or conceptual issues. In other words, these are the problems that developers face once they have started coding.

BIO: André Tost works as a solution architect in IBM's Software Group, where he focuses on SOA and Web services technology. He primarily engages with IBM's strategic business partners to enable them for J2EE and Web services. Before his current assignment, he spent many years in various IBM product development groups.

J-11 JCP Program: How the Java Technology Binary Software Standard is Managed and Evolves

Onno Kluyt, Sr. Director & Chair, JCP Program, Sun Microsystems

Thursday, Febuary 17, 2005 4:00 P.M. – 4:50 P.M.

Why a Java technology standard? Why technology communities? This session will explore the circle of adoption and business opportunity from an IT Manager and IT developer perspective. How Java technology fits into these circles, and the significance of conformance and the "Write Once, Run Anywhere" promise. The role the Java Community Process (JCP) program plays by carefully focusing on binary compatibility and bringing together the community to agree on standards and the results of this effort – multiple implementations from many sources based on Java technology.

BIO: Onno Kluyt is chair of Java Community Process (JCP) Program, Senior Director, JCP Program and Jini Program, Sun Microsystems. Onno leads the Program Office for the JCP, which oversees the process, manages its membership, guides specification leads and experts through the process, leads the Executive Committee meetings, and manages the JCP.org Web site. Onno also heads up the JINI program, including JINI technology and JINI community.

Attend a FREE* Tutorial

Tuesday, February 15, 2005, 8 – 11 a.m.

Developing Web Services Using Java Technologies

Aaron Williams

Aaron Williams, Manager, Java Community Process Program Office, Sun Microsystems

Why do developers favor Java technology for developing Web services? Java technology is the most ready platform for Web services and service-oriented architectures, complete with interoperability, platform independence, and security built in. This tutorial will review several Java technologies for Web services standards that have been developed or are currently being updated through the Java Community Process (JCP) program as JSRs – Java API for XML-based RPC (JAX-RPC), Java API for XML Binding (JAXB), Java API for XML Processing (JAXP), Java API for XML Registries (JAXR), JSR 109, Implementing Java Web Services, JSR 172, J2ME Web Services and JSR 173, The Streaming API for XML (StAX). Also to be highlighted: JSR 244, Java 2 Platform, Enterprise, Edition 5.0 Specification, JSR 181, Web Services Metadata for the Java Plaform, JSR 208, and Java Business Integration.

Sun microsystems

* Free Tutorials with VIP Preregistration ONLY!

VISIT **www.sys-con.com/edge** FOR TIMES & SCHEDULES

The program, including topics and times, is subject to change. Please refer to www.sys-con.com for all updates.

WEB SERVICES TRACK

WS-1 Ensuring Web Services Interoperability

Chris Ferris, IBM

Tuesday, Febuary 15, 2005 9:00 A.M. – 9:50 A.M.

Despite the open industry standards that underlie Web services, interoperability has been a key challenge for vendors and customers implementing Web services. One reason for this is that the relevant industry standards often permit multiple acceptable implementation alternatives. This presentation will discuss in detail the challenge of Web services interoperability and the role played by the premier industry organization formed to address it, the Web Services Interoperability Organization. In particular, the presentation will cover the critical importance of WS-I profiles to an organization's Web services initiatives, including the manner in which companies can put WS-I profiles immediately to work.

BIO: Chris Ferris is chair of the WS-I Basic Profile Working Group and a senior technical staff member with IBM's Emerging Technology Group. He has been actively engaged in open standards development for XML and Web services since 1999 and is an elected member of the OASIS Technical Advisory Board. Chris is also a coauthor and editor of the WS-Reliable Messaging specification.

WS-2 Web Services Standards: Going Behind the Mask

Glen Daniels, Sonic Software

Tuesday, Febuary 15, 2005 10:00 A.M. – 10:50 A.M.

Web services and service-oriented architectures (SOAs) are emerging as an integral part of the enterprise IT strategy. According to a recent IDC study, Web services – related revenue is expected to triple from $1.1 billion worldwide in 2003, to $3.4 billion in 2004, and $16.6 billion by 2008. As SOAs proliferate and the number of Web services added to them increases, standards will play an increasingly significant role. This session will look at the state of key Web services standards such as WS-Choreography, WS-Reliability and WS-ReliableMessaging, SOAP/MTOM/XOP, WSDL, XPath, XQuery, and WS-Notification as well as related Java standards and open source efforts. It will also look at the organizational impact of standards adoption in the industry.

BIO: Glen Daniels is manager of standards and consortia at Sonic Software and coauthor of *Building Web Services with Java.* He has been working with Web services technologies since their inception in the late '90s, and in addition to developing products and helping to found Apache's Axis project, he has been an active participant in standards bodies such as the W3C, and a member of the SOAPBuilders interoperability group.

WS-2B Solving Complex Business Problems Though SOA

John Daly, netNumina

Tuesday, Febuary 15, 2005 3:00 P.M. – 3:50 P.M.

WS-3 The XML Data Challenge

Nancy Vodicka, DataDirect Technologies

Tuesday, Febuary 15, 2005 4:00 P.M. – 4:50 P.M.

Most businesses store and query data with relational databases but need to use Extensible Markup Language (XML) to exchange and display data on the Web and with vendors and partners. As a result, programmers need to deal with both relational and XML data, often at the same time. Emerging standards such as XQuery, XQJ, and SQL/XML, promise to revolutionize data exchange and the ways applications are developed, deployed, and utilized. Learn the key facts about these standards, including what they mean, when they will be available, and what you, the developer, can do to prepare.

BIO: As the XML Product Manager at DataDirect Technologies, Nancy Vodicka is responsible for DataDirect Connect for SQL/XML, a database-independent SQL/XML implementation that is currently shipping, and DataDirect XQuery, a database-independent XQuery implementation that is currently in development. Nancy has more than 15 years experience in the software industry working with technologies such as XML, Web services, relational databases, and SQL.

WS-4 How To Bulletproof Your Web Services

David McCaws, Parasoft

Wednesday, Febuary 16, 2005 9:00 A.M. – 9:50 A.M.

Web services are gaining industry-wide acceptance and usage and are moving from proof-of-concept deployments to actual usage in mission-critical enterprise applications. Web services range from major services such as storage management and customer relationship management to much more limited services such as furnishing stock quotes or providing weather information. As companies and consumers begin to rely more and more on Web services, the need for developing reliable, high-quality Web services is even stronger. This session will explain issues specific to Web services and will illustrate solid engineering and testing practices required to ensure complete Web service functionality, interoperability, and security. Whether creating Web services from scratch or integrating a legacy back-end server via Web services, the practices and principles outlined in this session will be of great benefit.

BIO: David McCaw has over eight years of experience in helping software development teams improve quality throughout the development process. Over the last three years, he has led the Parasoft Web Services Solutions team, which has developed an industry-leading approach for Web services testing. He has implemented Web service quality solutions for development groups in organizations such as Sabre-Holdings, Yahoo! Overture, and McGraw-Hill. McCaw has an extensive background in the areas of Java and Web service reliability, performance, and security. He is involved with OASIS and WS-I, and is a frequent speaker at industry events.

WS-5 The Role of Policy in Web Services Integration – It's More Than Just Security

Toufic Boubez, Layer 7 Technologies

Wedneday, Febuary 16, 2005 10:00 A.M. – 10:50 A.M.

Too often today the preferences, terms, and conditions describing how a Web service behaves when discovered and invoked is programmed right into the business logic. Hard-coding this behavior logic however introduces cost, complexity, and rigidity into a Web services architecture. A better approach is to abstract a Web services usage "policy" out of code where this metadata can be managed as need be. This session introduces the concept of Web Services Policy and describes how the construct can be used to implement a more customized and versatile Web service infrastructure.

BIO: Toufic Boubez is a well-respected and renowned Web services visionary. Prior to cofounding Layer 7 Technologies, Toufic was the chief Web services architect for IBM's Software Group and drove their early XML and Web services strategies. He is a sought-after presenter and has chaired many XML and Web services conferences. He is an author of many publications and his most recent book is the top-selling *Building Web Services with Java: Making Sense of XML, SOAP, WSDL ,and UDDI.*

WS-6 B2B Policy Enforcement: The Third Rail of SOA Implementation

Alistair Farquharson, Digital Evolution

Wednesday, Febuary 16, 2005 4:00 P.M. – 4:50 P.M.

One of the great benefits of a service-oriented architecture is the ability it gives you to extend programmatic, integration capabilities to business part-

VISIT **www.sys-con.com/edge** FOR TIMES & SCHEDULES

The program, including topics and times, is subject to change. Please refer to www.sys-con.com for all updates.

ners. Going beyond simple sharing of data with partners, SOA enables true B2B application integration. At the same time, this capability creates a vexing security policy enforcement dilemma. How can you be sure that a user from a partner organization is actually authorized to integrate with your applications? How can you authenticate that user? Do you even want that headache in the first place? This session will discuss the issues that arise in B2B security policy enforcement and explore several proven approaches to solving the problem. In particular, it will focus on the emerging technology of XML Virtual Private Networks (XML-VPNs) and their potential to mitigate security policy enforcement issues in B2B SOA implementations.

BIO: Alistair Farquharson is the CTO of Digital Evolution, where he spearheads product development and provides thought leadership to enterprise customers implementing Web services. His skills span many industries and include designing and implementing system architectures, as well as spearheading initiatives such as development/team lead. He is an expert in custom-application development, distributed environments, architecting scalable hardware and software applications and systems, and Web services application development.

WS-7 Driving SOA Governance

Brent Carlson, LogicLibrary

Wednesday, Febuary 16, 2005 5:00 P.M. – 5:50 P.M.

In the past year, Web services and service-oriented architectures (SOAs) have become mainstream because of their ability to provide business agility and flexibility through integration, productivity, and reuse. With SOA enablement on the rise, IT groups must adopt SOA governance as a means of controlling what and how services located within an SOA are deployed. This session will discuss SOA governance, specifically how an organization can manage and control assets and artifacts located within an enterprise, while ensuring that deployed assets meet an organization's business and technical architectural standards. It will also outline governance best practices such as monitoring the UDDI publish process in order to seamlessly tie together the development and operational views of Web services within the enterprise.

BIO: Brent Carlson drives the development and delivery of LogicLibrary's products. He is a 17-year veteran of IBM, where he served as lead architect for the WebSphere Business Components project and held numerous leadership roles on the "IBM San Francisco Project." He is a member of the Eclipse Board of Stewards and a BEA Regional Director.

WS-8 SOA: From Pattern to Production

David Chappell, Sonic Software

Thursday, Febuary 17, 2005 9:00 A.M. – 9:50 A.M.

Service-oriented architecture (SOA) represents the opportunity to achieve broad-scale interoperability, while providing the flexibility required to continually adapt technology to business requirements. No small feat, particularly when one considers the extent and complexity of today's IT environments. As both a technology concept and IT discipline, the challenge inherent in SOAs is maintaining the right architectural approach. If all services in an SOA are treated as interdependent point-to-point interfaces, then the complexity of implementing and maintaining them in this spaghetti-like architecture becomes enormous. The enterprise service bus (ESB) has emerged as one of the first true SOA product offerings, bringing SOA from pattern to production. ESBs provide a framework for building and deploying an event-driven, enterprise SOA and accommodates the configuration, hosting, and management of integration components as services across the business.

BIO: VP and chief technology evangelist for Sonic Software, Dave Chappell has over 18 years of experience in the software industry covering a broad range of roles including R&D, codeslinger, sales, support, and marketing. He also has extensive experience in distributed computing, including message-oriented middleware, CORBA, COM, and Web application server infrastructure.

WS-9 High Performance Web Services – Tackling Scalability and Speed

Sameer Tyagi, Sun Microsystems

Thursday, Febuary 17, 2005 10:00 A.M. – 10:50 A.M.

Web services facilitate application-to-application integration and interoperability across different platforms. However, critics usually point to an inefficient processing model and bandwidth requirements for developing Web services. This is often cited as a reason why Web services cannot perform and scale well in production environments. This session takes a detailed look at performance and scalability issues around Web services in the real world, as well as strategies that architects and developers can adopt to mitigate such risks in these applications. Some analytical and modeling strategies that enable acceptable application performance will also be covered.

BIO: Sameer Tyagi works as a senior Java architect with Sun Microsystems. He remains focused on architecture, design, and implementation of large-scale enterprise applications with Java technology. His publications include industry periodicals and books on Java and J2EE technologies including *Java Web Services Architecture.*

WS-10 So You Want an SOA: Best Practices for Migrating Toward Service Orientation in the Enterprise

Eric Newcomer, IONA

Thursday, Febuary 17, 2005 3:00 P.M. – 3:50 P.M.

Replacing complex, monolithic applications with nimble applications built from exposed services promises increased developer productivity, greater flexibility, and ultimately reduced cost. The adoption of Web services and SOA can also remove a significant level of complexity and integration problems from enterprise application development projects. But, as with any large-scale project, IT departments must have the right plan and the right resources in place to ensure a successful transformation of their computing infrastructure. This article will explore what IT organizations need to know to be successful in their attempts to migrate the enterprise to a service-oriented architecture.

BIO: In the role of chief technology officer at IONA, Eric Newcomer is responsible for IONA's technology roadmap and the direction of IONA's e-business platforms as relates to standards adoption, architecture, and product design.

WS-11 Four Abilities SOA Will Lack Without a Registry

Luc Clement, Systinet

Thursday, Febuary 17, 2005 4:00 P.M. – 4:50 P.M.

A service-oriented architecture (SOA) is the design blueprint for seamless connectivity between business processes and IT infrastructure, enabling innovation and improving productivity. SOA provides the most efficient, standard way to dynamically interoperate with any customer, supplier, product or employee. SOA makes integration intrinsic. Web services are the foundation building blocks of an SOA, and they are already proliferating inside most enterprises. In an SOA, Web services become business services with the ability to perform a particular function or access data dynamically. This presentation will discuss the four abilities that a registry provides for an SOA.

BIO: Luc Clement is director of product marketing, SOA Registry for Systinet. He is also cochair for the UDDI Specification Technical Committee. Formerly Microsoft UDDI Program Manager, Luc is well known in the UDDI community and has been heavily involved with the UDDI specification for several years.

Register by
JANUARY 21, 2005
SAVE UP TO $300
Register Today!
www.sys-con.com/edge2005

VISIT **www.sys-con.com/edge** FOR TIMES & SCHEDULES

The program, including topics and times, is subject to change. Please refer to www.sys-con.com for all updates.

SPECIAL INSERT: WEB SERVICES EDGE 2005
WWW.SYS-CON.COM/EDGE
FEBRUARY 15-17, 2005
JOHN B. HYNES CONVENTION CENTER • BOSTON, MA

International Web Services Conference & Expo

Register by JANUARY 21, 2005 SAVE Up To $300

For Assistance Tel: 201-802-3066 Fax: 201-782-9601

CONFERENCE & EXPO: February 15 – 17, 2005

John B. Hynes Veteran Memorial Convention Center • Boston, MA

THREE WAYS TO REGISTER FOR CONFERENCE

1) **On the Web**: Credit Cards or "Bill Me" Please make checks payable to SYS-CON Events
2) **By Fax:** Credit Cards or "Bill Me" 201-782-9601
3) **By Mail:** 135 Chestnut Ridge Road, Montvale, New Jersey 07645, Attention: Registration

Please note: Registrations are not confirmed until payment is received.

Please complete sections 1, 2, 3 and 4

1 YOUR INFORMATION (Please Print) ☐ Mr. ☐ Ms. ☐ Mrs.

First Name ______________________ Last Name ______________________
Title ______________________
Company ______________________
Street ______________________
Mail Stop ______________________
City ______________________
State ______________________ Zip ______________ Country ______________
Phone ______________________
Fax ______________________ E-Mail ______________________

2 PAYMENT METHOD: (Payment in full due with registration)

☐ Check or Money Order Enclosed (Registration confirmed upon receipt of payment)

Check # ______________________ Amount of Check $ ____________

Charge my ☐ Visa ☐ MasterCard ☐ American Express ☐ Discover

Name on card ______________________

Card # ______________________ Exp. Date ____________

Signature ______________________

Billing Address (if different from mailing address)
____

3 PLEASE INDICATE YOUR CONFERENCE CHOICE:

Total Registration fee $____________

On Site

		Before 12/31/04	Before 1/21/05	Before 2/11/05
❑ **3D: Three Day Conference** Good for all three days of the conference program, including keynotes, all conference sessions.	$1,195.00	$1,395.00	$1,495.00	$1,695.00
❑ **2D: Two Day Conference** Good for two days of the conference program, including keynotes, all conference sessions.	$995.00	$1,195.00	$1,295.00	$1,395.00
❑ **1D: One Day Conference** Good for one day of the conference program, including keynotes, all conference sessions.	$595.00	$695.00	$695.00	$795.00
❑ **VIP PASS** FREE Tutorials with VIP preregistration ❑ FREE - Novell Tutorial (Feb. 15) ❑ FREE - Developing Web Services Using Java Technology (Feb. 15) ❑ FREE - The Next Generation of Visual Studio (Feb. 16) ❑ FREE - Security, The New Reality (Feb. 17)	FREE	FREE	FREE	$75.00
❑ **EO: Expo Hall Only** Includes keynotes, expo presentations, and workshops	FREE	FREE	FREE	$50.00

SPECIAL GROUP RATES
Registering a group of four or more? Please call Jim Hanchrow at 201-802-3066, and receive an additional discount.

4

A. Your Job Title
❑ Senior Software Developer/Engineer/Architect/Analyst
❑ Director of IT/Development/Engineering
❑ Software Developer/Engineer/Architect
❑ Systems Architect/Engineer/Analyst
❑ Software Consultant/Analyst
❑ Project Manager/Team Leader
❑ Business Development
❑ General Manager/Division or Department Head
❑ CTO,CIO, Chief Architect
❑ CEO, President, Owner, Partner/Principal
❑ VP/Technical Director
❑ Network Administration/Manager/Consultant
❑ Web Developer/Programmer
❑ Database Administrator
❑ VP/Sales & Marketing Director /Product Management
❑ Other (please specify)______________

B. Business/Industry
❑ Agriculture/Mining/Oil/Gas
❑ Computer Hardware and Electronics
❑ Construction/Architecture/Engineering
❑ Consulting (non-computer)
❑ Education
❑ Banking/Insurance/Accounting
❑ Government (including Military)
❑ Healthcare/Pharmaceuticals/Biotech/Biomedical
❑ Internet/Web/e-Commerce
❑ Manufacturing and Process (other than computer related)
❑ Media/Marketing/Advertising
❑ Non-Profit/Trade Association
❑ Real Estate/Legal
❑ Telecommunications
❑ Transportation/Rail/Sea/Air/Ground
❑ Travel/Hospitality/Entertainment
❑ Utilities
❑ Wholesale/Trade/Distribution/Retail
❑ Other (please specify)
____

C. Total Number of Employees at Your Location and Entire Organization (check all users that apply):

	Location	Company
10,000 or more	01 ❑	01 ❑
5,000 - 9,999	02 ❑	02 ❑
1,000 - 4,999	03 ❑	03 ❑
500 - 999	04 ❑	04 ❑
100-499	05 ❑	05 ❑
100 or less	06 ❑	06 ❑

D. Please indicate the value of software products and services that you recommend, buy, specify or approve over the course of one year:
❑ $10 million or more
❑ $1 million - $9.9 million
❑ $500,000 - $999,999
❑ $100,000 - $499,999
❑ $10,000 - $99,999
❑ Less than $10,000
❑ Don't know

E. What is your company's gross annual revenue?
❑ $10 billion or more
❑ $1 billion - $9.9 billion
❑ $100 million - $999 million
❑ $10 million - $99.9 million
❑ $1 million - $9.9 million
❑ Less than $1 million
❑ Don't know

F. Do you recommend, specify, evaluate, approve or purchase mainframe products or services for your organization?
01 ❑ Yes 02 ❑ No

G. Which of the following products, services, and/or technologies do you currently approve, specify or recommend the purchase of?
❑ Application Servers
❑ Web Servers
❑ Server Side Hardware
❑ Client Side Hardware
❑ Wireless Device Hardware
❑ Databases
❑ Java IDEs
❑ Class Libraries
❑ Software Testing Tools
❑ Web Testing Tools
❑ Modeling Tools
❑ Team Development Tools
❑ Installation Tools
❑ Frameworks
❑ Database Access Tools / JDBC Devices
❑ Application Integration Tools
❑ Enterprise Development Tool Suites
❑ Messaging Tools
❑ Reporting Tools
❑ Debugging Tools
❑ Virtual Machines
❑ Wireless Development Tools
❑ XML Tools
❑ Web Services Development Toolkits
❑ Professional Training Services
❑ Other [Please Specify]
____

SYS-CON EVENTS

If you require special assistance covered under the Americans with Disabilities Act, please call 201-802-3066 by February 4, 2005.

CANCELLATIONS, SUBSTITUTIONS, REFUNDS
Fax written request to SYS-CON Registration 201-782-9601. Requests for refunds received prior to January 7, 2005 will be honored, less a 10% handling charge; requests received after January 7, 2005, and before February 4, 2005, will be honored less a 20% handling charge. No requests for refunds will be honored after February 4, 2005. Requests for substitutions must be made in writing prior to February 11, 2005. No one under 18 is permitted to attend. No warranties are made regarding the content of sessions or materials.

Speakers, sessions and schedule are subject to change without prior notice.

No solicitation by anyone other than official exhibitors, sponsors or marketing partners is permitted. Such behavior is cause for expulsion without refund.

Lazy Programmers Can Make Cache Today

by Warren MacEvoy

Let me begin with a philosophical rant. There is a motto from scientific computing that carries over to many areas of computer science:

The gains made by better algorithms almost always outstrip the gains from better hardware.

I've frequently seen where algorithm improvements pay by factors of ten to tens of thousands in CPU time. One change I made in a numerical algorithm improved CPU requirements by a factor of 50,000: from weeks on a super computer to minutes on a workstation.

Any business-savvy engineer knows that algorithm improvements come at a price: the engineer's time. Striking that balance makes software systems move forward rather than stagger to a halt in bloat and dysfunction. It also helps to use people who actually know what they are doing – knowing how to compile code doesn't make you a software engineer any more than knowing how to spell makes you a writer. End of rant.

On to (rant-related) business. On most Web sites, think of how many times a data source will be used to retrieve the same data and produce the same content over and over again. Most successful services deliver a highly redundant amount of information to their users. For example, the *JDJ* Web site will deliver this (same) content to perhaps a hundred thousand users. If the servers are overtaxed, customers will experience significant delays or malfunctions.

There are several useful solutions to this. Well-configured caching proxy servers come to mind, although server-side scripting makes this difficult. Buying more hardware will eventually fix the problem, which may be the correct business solution.

But what about asking programmers to be a little more lazy?

The source for the LazyFileOutputStream can be downloaded from http://bpp.sourceforge.net/download/bpp-0.8.5b/src/bpp/LazyFileOutputStream.java. It acts just like a regular FileOutputStream except that if created on a file that already exists, it "reads" the data from the file instead of writing it. The stream compares what is already in the file with what you are currently writing to it. If at any point it sees that there is a difference in the data you are writing this time compared to what is already there, the stream automatically switches to a write mode that writes over the remainder of the file with the changes.

The upshot is, if your program generates the same output twice, the output file is unmodified the second time (leaving the original modification date). First, by simply changing FileOutputStream to LazyFileOutputStream, any downstream processing can use timestamp information on the files to check if they need to do anything at all. If the timestamp hasn't changed, neither have the contents.

But wait, there's more! In addition to the standard close(), the LazyFileOutputStream also supports abort(). This method effectively states "I'm done now, leave the rest of the file alone." The remainder of the file will be the same, even without reproducing it. This means that if you determine at an early point in the processing of the file that it's going to turn out the same, you can simply abort() to leave it alone. It's similar to the idea of not changing the modification dates on files that are rewritten with the same data, but allows you to save CPU time for the current process step as well as downstream processing.

Certain engines produce part of the template before you can conveniently intervene to decide if you really need to regenerate it. By opening up the output as a Lazy file, you can just abort() early and have the old version with the the old modification time around for downstream processing.

Okay, rant concluded and point made: CPUs around the planet are spinning through the same data tens of thousands of times, producing the same content tens of thousands of times. Instead of buying great big servers to manage this, a smart caching policy based on lazy file writers and some modification time testing could save some sites that same wild-sounding factor of 50,000, without having to buy 50,000 new servers.

Anecdote #1

There is a certain technical advantage to this style of writing data as well: most storage devices are easier to read from than write to, adhering to the 80/20 rule – 80% of file access will be reads, 20% will be writes. The LazyFileOutputStream takes advantage of that for the many files that are simply rewritten with the same content.

Anecdote #2

There must be a few curled toes out there saying to themselves, "Why not LazyFileWriter?" There are good technical reasons for the OutputStream: the logic of the data written must be checked in its raw "byte" format for the idea to work correctly, and you can always wrap this in an OutputStreamWriter followed by a BufferedWriter, which is what I recommend.

Now I'm even done with the anecdotes. Have a nice day.

Warren D. MacEvoy is an assistant professor of computer science in the department of computer science, mathematics, and statistics at Mesa State College, Grand Junction, Colorado.

wmacevoy@mesastate.edu

ENTERPRISE CHARTING SOLUTIONS

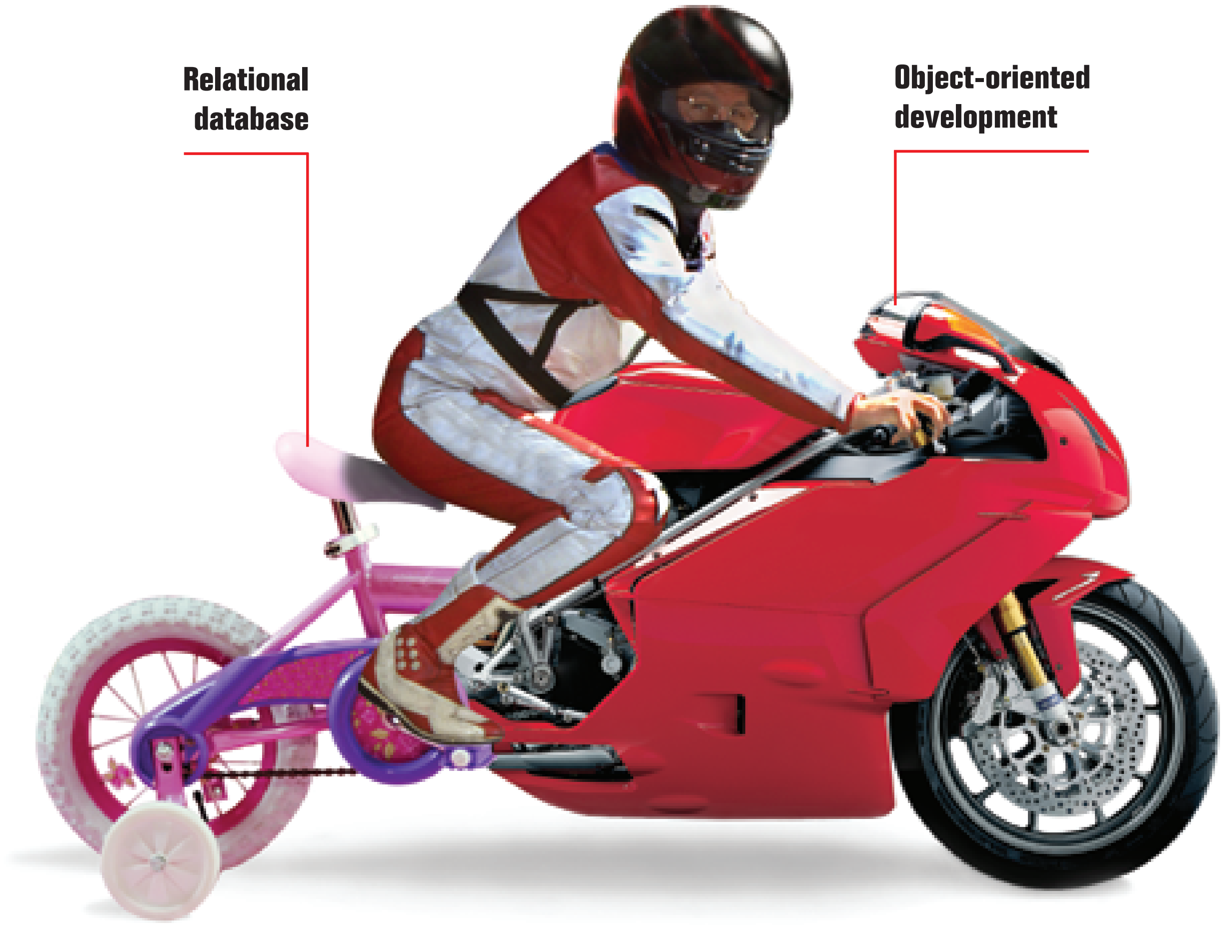
Need More Than A Pretty Face?

For Serious Enterprise-Level Data Visualization & Analysis

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

Chart FX

www.softwarefx.com



A BETTER DATABASE CAN SPEED UP YOUR DEVELOPMENT CYCLE

If your relational database isn't a good match for your object-oriented development, you need a new database.

Caché, the *post-relational* database from InterSystems, combines high-performance SQL for faster queries <u>and</u> an advanced object database for rapidly storing and accessing objects. With Caché, no mapping is required between object and relational views of data. Every Caché class can be automatically projected as Java classes or EJB components with bean-managed persistence. Plus, every object class is instantly accessible as tables via ODBC and JDBC.

That means huge savings in both development and processing time. Applications built on Caché are massively scalable and lightning fast. They require little or no database administration. And Caché's powerful Web application development environment dramatically reduces the time to build and modify applications.

We are InterSystems, a specialist in data management technology for over twenty-six years. We provide 24x7 support to four million users in 88 countries. Caché powers enterprise applications in healthcare, financial services, government, and many other sectors. Caché is available for Windows, OpenVMS, MAC OS X, Linux, and major UNIX platforms – and it is deployed on systems ranging from two to over 50,000 simultaneous users.

InterSystems CACHÉ™
Make Applications Faster

Try a better database. For free.

Download a free, fully functional, non-expiring copy of Caché or request it on CD at www.InterSystems.com/match3