Order a Solutions CD
www.rational.com/offer/xdecd Try Rational® XDE™ | Rational® the software development company

JAVA™ DEVELOPER'S JOURNAL

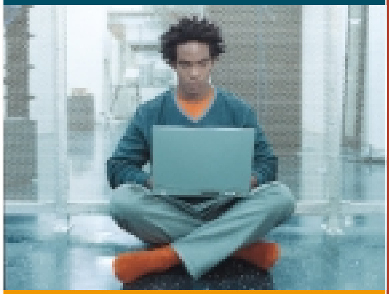
The World's Leading Java Resource

April 2002 Volume: 7 Issue: 4

JAVADEVELOPERSJOURNAL.COM

be liberated

Order a Solutions CD
www.rational.com/offer/xdecd



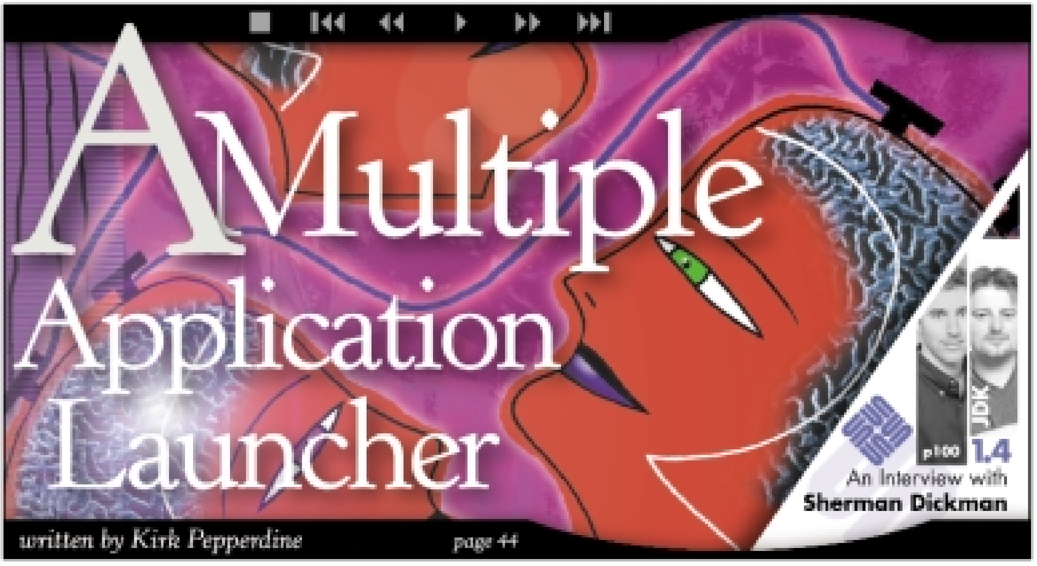
Rational® XDE™

Full Conference Program
INSIDE pg. 107

THE LARGEST WEB SERVICES CONFERENCE & EXPO IN THE WORLD!
web services EDGE
JDJEDGE
XMLEDGE
Conference: June 24-27 Expo: June 25-27
SAVE $200.00
Call Today!
The Future of Enterprise Computing
JACOB JAVITS CONVENTION CENTER

RETAILERS PLEASE DISPLAY
UNTIL JUNE 30, 2002
$5.99US $6.99CAN

SYS-CON MEDIA



A Multiple Application Launcher
written by Kirk Pepperdine page 44

JDK 1.4 An Interview with Sherman Dickman

Sonic Software

www.sonicsoftware.com

Zero G

www.zerog.com

BEA

www.bea.com/download

JAVA DEVELOPER'S JOURNAL

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

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

WRITERS IN THIS ISSUE
BILL BALOGLU, JASON R. BRIGGS, KEITH BROWN, JURIC B. MATJAZ, LARRY MCCAY, JAMES MCGOVERN, BILLY PALMIERI, KIRK PEPPERDINE, DAN PILONE, IVAN ROZMAN, AJIT SAGAR, MICAH SILVERMAN, SCOTT SILVERMAN, JOHN WALKER, JAMES WHITE, ALAN WILLIAMSON, BLAIR WYMAN

SUBSCRIPTIONS:
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS, PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT
SUBSCRIPTION HOTLINE: SUBSCRIBE@SYS-CON.COM
COVER PRICE: $5.99/ISSUE
DOMESTIC: $49.99/YR. (12 ISSUES)
CANADA/MEXICO: $79.99/YR. OVERSEAS: $99.99/YR.
(U.S. BANKS OR MONEY ORDERS). BACK ISSUES: $10/EA., INTERNATIONAL $15/EA.

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

ALAN WILLIAMSON EDITOR-IN-CHIEF

There May Be Trouble Ahead . . .

As Nat King Cole famously sang, we have to "face the music and dance…" This month's editorial is coming to you with a *reader beware* warning!

I've been engaged in some great debates over the last month on a variety of topics, but the one that has caught my interest is the old chestnut regarding the longevity of Java. Is it here to stay? If not, how long do we have? Quite rightly, it's being talked about and I've had the good fortune to brush shoulders with a number of big names in our industry who have given me their perspectives on the whole debate. I have my own feelings about where Java is headed and I do believe that if, as a community, we don't get our act together, we may have only five years left at the most. After talking to my counterparts, it would appear I'm being overly generous with five years.

What's happening? Well, it's our old friend C# and its relentless march toward the development community. Setting aside the old argument that due to Microsoft's dominance it may well win the day, it's interesting to look at other reasons why C# may win the battle. Let's blow away some misconceptions that you may or may not be aware of regarding this new kid.

Myth #1: C# is a Windows-only technology.

You could be excused for believing that, but did you know there's a major movement in the open source world to port the CLR (Common Language Runtime, i.e., their JVM!) to operating systems other than MS Windows? Linux, to name one. Imagine for a moment being able to run your .NET services alongside Apache on a Redhat box, seamlessly integrating into the rest of the network. This alone would be a major blow to server-side Java. It's also a subtle way for Microsoft to unofficially support the growing number of Linux seats without losing face (read www.halcyonsoft.com/news/iNET_PR.asp).

Myth #2: C# is an inferior Java clone.

This is the most dangerous one and the one you probably tell yourself in order to keep the scales tipped in Java's favor. The truth is, it's not an inferior clone; it's a different clone, with many arguing that the differences are minute to the majority of the developer community. It will be frighteningly easy for Java developers to move over to C# with no real headaches to contend with. I suspect this was always on Microsoft's mind when developing the language (read www.prism.gatech.edu/~gte855q/CsharpVsJava.html).

Myth #3: C# is for developing Web services only.

Most definitely not, and I have heard this one retorted back to me on a number of occasions. Ironically, this is the one area that could really hurt Java – on the client. As you know, Java has not made any significant headway in this space due mainly to its awfully slow Swing implementation. While the recent release of JDK1.4 has brought significant performance gains, it's still nowhere near the speed of its native Windows applications with respect to fast, snappy responses (although it must be said, the speed of a Swing application on a Mac OS-X does show what could be achieved). C# is the new building block for Windows applications, the next VB! And we know how many applications popped up when VB hit the market (read www.c-sharpcorner.com/WinForms.asp).

Okay, how many of you think I've abandoned all hope for Java and have gone to the dark side? I suspect some of you are questioning my loyalties at this precise moment, wondering if I'm fit to occupy my role as EIC. Well, don't panic, I'm merely being a realist and looking at it from all angles. You'd be the

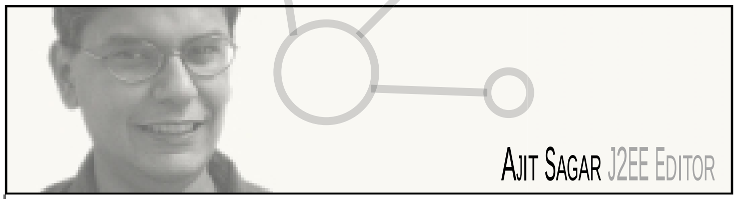
–continued on page 98

alan@sys-con.com

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

TogetherSoft Corporation

www.togethersoft.com/1/jd.jsp



AJIT SAGAR J2EE EDITOR

Designing for the *n*th Tier

You want to develop a new business application based on your particular business problem. You get a software team to pull together the right mix of technologies to build the required software components. You choose an architect to capture your business requirements and to define the right mix of software and hardware to deploy the appropriate solution. You put together a development team to implement the business processes of your application in the form of software components that magically do everything. Now all you have to do is sell the solution. Right?

Wrong. Nothing is ever that easy, especially in software development. One of the toughest challenges when designing a software application is to define the deployment architecture that ultimately hosts the application's components. Unfortunately, in an application development cycle it's typical to postpone this until the software components are near completion. For successful businesses, it's imperative that the software architecture be well defined ahead of time. Otherwise, changing the developed components to adapt to the runtime architecture is a very costly proposition in both time and money.

The Java platform was designed to support distributed applications from the get-go, although the pieces to effectively do so came later. When Sun divided Java into three editions, it was clear that the platform had come full circle, with J2EE as the architecture for developing business applications across the different tiers. During the past two years, the vendors have solidified their offerings for building applications based on Java's distributed technologies and encompassed by J2EE. However, your software architect has a plethora of choices for developing the end application. How well your architect accomplishes this is based on his or her experience in identifying the appropriate technologies from the wide gamut offered by J2EE. This, in turn, is based on the ability of your software architect to identify the deployment architecture needed to finally host the application.

The current release of the platform offers several options on how to structure the tiers of your application. Specifically, the maturation of the EJB component model, the support around XML and messaging, the introduction of the JCA architecture, and the value-added support offered by the leading application server vendors provide a very stable environment for defining the deployment architecture at an early stage. In addition, the mismatch between the requirements definition and analysis tools such as UML and software design environments has decreased tremendously. Most of the leading application servers and IDE environments are moving toward offering consolidated environments that allow requirements capture and analysis and software development using the same suite of tools. One of the technologies that makes this possible is XML, since it provides a uniform mechanism for exchanging data between environments.

However, getting the right set of tools doesn't complete the job; they have to be applied properly. Good architectural design for an *n*-tier application requires the ability to select and eliminate the features offered by the platform and supporting environments to maintain the needs of a specific application. Several application development efforts are based on the examples that are bundled with development tools or offered by the platform.

A case in point in the J2EE world is Sun's Pet Store example. While this is supported by most of the leading vendors, it's too trivial for designing a real-world application. Ultimately, you'll have to tailor the technologies to your specific business requirements. This involves answering hard questions like: Should you develop your application on a three-tier architecture using only the Web component layer as opposed to the EJB layer provided by J2EE? When should you use direct JDBC instead of container-managed database access? Are there cases that warrant the use of a two-tier model? Can you base your application on a pure asynchronous messaging layer instead of an EJB-based middle-tier?

The answer to all these questions is: yes, but not always. Fortunately, several J2EE design patterns that embody the experience of millions of J2EE developers are available for making the correct design choices.

ajit@sys-con.com

AUTHOR BIO

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

J2EE INDEX

J2EE for EAI

Transactions, security, naming, performance, and Web services

WRITTEN BY
Matjaz B. Juric
& Ivan Rozman

Last month's article "J2EE As the Platform for EAI" (*JDJ*, Vol. 7 issue 3) discussed the suitability of the J2EE platform for EAI (Enterprise Application Integration). This article addresses more advanced integration topics, particularly transaction and security, support for Web services, and an overview of J2EE application servers.

EAI enables concurrent access to data and functionality shared between disparate existing applications while maintaining integrity, consistency, performance, and recoverability. Transactions play an important role in ensuring integrity, consistency, and recoverability.

Support for Transactions

When talking about integration we have to support transactions that span existing applications, databases, and newly developed application components. In general, the following are the most frequently seen scenarios:

- A virtual component or a wrapper in the business component tier accesses several different databases (in the EIS tier) within a single transaction.
- Several virtual components are included within a transaction and access several different databases within a single transaction (see Figure 1).
- A virtual component calls different EIS systems within a transaction.
- Several virtual components deployed in different containers call different EIS systems within a transaction (see Figure 2).
- A combination of the above – several virtual components may call a combination of EIS systems and access several different databases within a single transaction.

For each scenario it's necessary to figure out how transactions can be propagated from the J2EE to the existing systems and back. Things get more complicated because some of the existing databases and applications may already provide transactional support.

Local and Distributed Transactions

In general, a transaction includes several resources (such as components, databases, and existing systems). Each resource taking part in a transaction needs a resource manager, which manages access to one or more resources. The whole transaction, however, is managed by the transaction manager. Figure 3 illustrates these relationships.

The transaction manager coordinates different resource managers. Therefore, it has to establish and maintain the state of the transaction. This state is managed by the transactional context, which associates the transactional operations on the resources. All resources that participate in a transaction share the same transactional context. Usually the transaction manager also takes care of propagating the transactional context between the resource managers. Resource managers, however, have to notify the transaction manager about their participation and the current state of the transaction. Based on this information, the transaction manager decides whether to commit or roll back the transaction. In a commit or rollback, all resource managers take part.

In simple applications that span only a single resource (a database, for example) we need only one resource manager. We call such relatively simple transactions *local* transactions. Unfortunately, in complex integration scenarios we usually aren't that lucky.

We have to use transactions that span different systems, as shown in Figure 3. These are called *distributed transactions*. Such transactions are more complex than local ones and require a special algorithm, the two-phase commit, to commit them. It performs the voting phase and the commit phase in a complicated sequence of interactions between the transaction manager and resource managers. Fortunately, there are two standards for distributed transactions that enable transactional interoperability between different products:

- ISO TP model
- X/Open DTP (Distributed Transaction Processing) model

The ISO TP model doesn't have many supporters and therefore doesn't play an important role in distributed transaction processing. Far more important is the X/Open DTP Model, which has established itself as a standard among vendors providing transactional systems. X/Open DTP specifies two interfaces between the major elements of a transactional system:

- ***TX interface:*** The interface between application components and the transaction manager, it enables an application component to join a transaction.
- ***XA interface:*** This defines the interac-

Metrowerks Corp.

www.wireless-studio.com

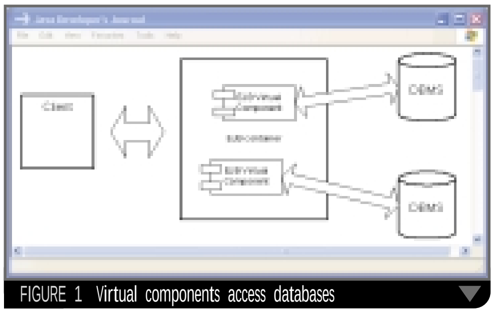


FIGURE 1 Virtual components access databases

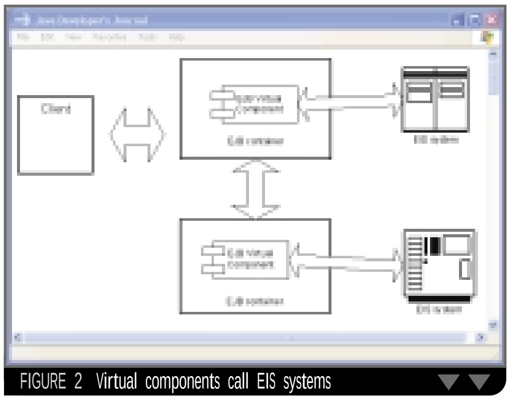


FIGURE 2 Virtual components call EIS systems

tion between the resource managers and the transaction managers. This interface enables the cooperation of different resource managers, implemented by different resources, to take part in a single distributed transaction

Most widely implemented commercial products support the X/Open DTP standard, including:

- DBMSs such as Oracle, Informix, Sybase, and MS SQL Server
- TP monitors such as Encina, Tuxedo, and TopEnd
- MOMs such as IBM MQSeries and MS MQ
- Distributed-component models such as CORBA (through CORBA Object Transaction Service)

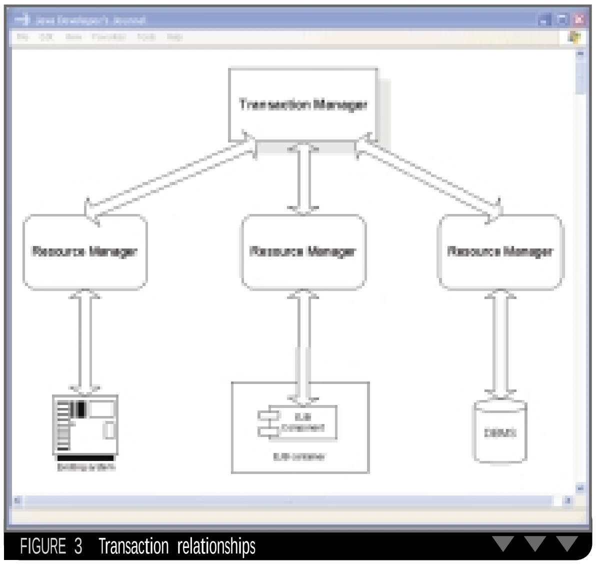


FIGURE 3 Transaction relationships

How J2EE Supports Transactions

You probably know that J2EE supports transactions in the Web component tier, the business logic tier, and the EIS tier. The J2EE specification doesn't require support for transactions on the client tier. Although J2EE supports transactions in the Web component tier, it's a much better idea to delegate the transactional work to the business logic tier.

J2EE also provides two ways to specify transactions. It supports programmatic and declarative transactions. With declarative transaction demarcation, the J2EE components are marked as transactional at deployment. The container (the EJB container, for example) is responsible for determining transaction boundaries.

With programmatic transaction demarcation, the transaction clients demarcate transaction boundaries themselves. In most cases, the transaction clients associate the transactional context with the thread, which executes the transactional operations.

To access the transaction service functionality, J2EE has the JTA (Java Transaction API), which provides an abstraction layer on top of the JTS (Java Transaction Service). JTA specifies standard interfaces through which different resources participating in a transaction can communicate. The abstraction layer provided by the JTA enables the platform to choose which JTS implementation it will use. The implementation, which is typically provided by the application server, is transparent to the application components because the application components don't interact with the JTS directly.

The really important fact for integration is that the JTS is compliant with the CORBA Object Transaction Service (OTS) 1.1. More precisely, it's the mapping of the CORBA OTS to Java. It propagates transactions using the IIOP protocol. As I've already mentioned, CORBA OTS provides support for distributed transactions, so we can infer that the JTS also supports them.

However, the J2EE specification 1.3 doesn't *require* a J2EE implementation to support distributed transactions. Fortunately, most widely implemented J2EE application servers, such as BEA WebLogic, IBM WebSphere, and Oracle 9*i*AS, already provide support for these transactions.

Existing Systems and Transactions

Support for distributed transactions is thus required for serious integration of transactional behavior between the J2EE and existing transactional EIS systems that are X/Open DTP compliant; fortunately, most EIS systems are compliant. When we access such systems within transactions, they behave in accordance with other resources included in the transaction. They commit changes only if the whole transaction is committed; otherwise they roll back the results. Within such a transaction we can interoperate interactions with several EIS systems. The coordination and the propagation of transactional context is handled automatically.

EIS systems that implement local transactions that aren't X/Open DTP compliant require more manual work to achieve transaction integration. Such transactions are managed by the resource manager of the underlying EIS; therefore the J2EE application server won't be aware of them and won't be able to control or influence them. With local transactions we have to commit or roll back each system explicitly and manually.

A technique to manage this scenario is to use compensating transactions. For every system in which we use a local transaction, we have to define a compensating transaction with which we can undo the effects of a previously committed local transaction. Compensating transactions have to be implemented programmatically and they introduce a few problems:

- They don't provide high enough isolation for clients accessing the EIS system. The transaction is committed and then reversed after some time. In the meantime, other clients can see the inconsistent state. Compensation transactions implement only the read uncommitted isolation level.
- If the compensating transaction isn't successful, the state of the EIS system can become inconsistent.

We have to be aware of these problems and provide application logic that will resolve these situations if they arise.

Support for Security

Security is another important topic for EAI. When defining the integration architecture, we have to ensure that existing applications remain secure and that we don't re-create functionality that already exists. However, we also have to ensure the integration of security mechanisms. Imagine an integrated information system in which each user has to use an application-specific way to authenticate and authorize. Using such

DataDirect Technologies

www.datadirect-technologies.com

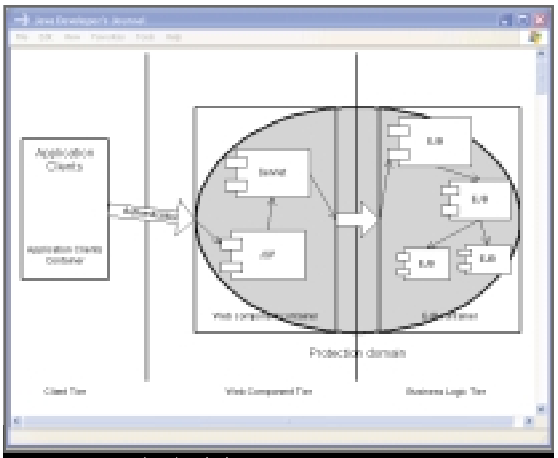
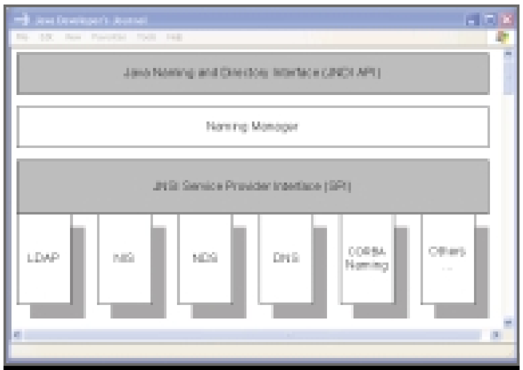


FIGURE 4 Protection domain in J2EE



Java Naming and Directory Interface (JNDI API)
Naming Manager
JNDI Service Provider Interface (SPI)
LDAP
NIS
NDS
DNS
CORBA Naming
Others ...

FIGURE 5 JNDI architecture

a system would be pretty tedious, requiring users to log on several times using different username and password combinations.

What we need is a way to propagate security contexts through integrated applications. In other words, we need a standardized security model that can handle a wide range of applications, establish an end-to-end security through the tiers, and provide a single authentication point for the user.

As we know, Java has a policy-driven, domain-based security model. Java provides extension packages that were initially introduced as optional but have been integrated into J2SE 1.4, including:

- ***Java Authentication and Authorization Service (JAAS):*** Provides a framework for authentication and authorization. J2EE 1.3 requires that J2EE application servers use JAAS in Web component and EJB containers, particularly to support J2EE connector architecture.
- ***Java Secure Socket Extension (JSSE):*** Provides packages for enabling secure communications using SSL and TLS protocols and includes functionality for encryption, authentication, and message integrity.
- ***Java Cryptography Extension (JCE):*** Provides a framework and implementation for encryption, key generation, key agreement, and MAC (Message Authentication Code) algorithms.

Authentication is often performed in two steps. First, based on a password or some other secret information, an authentication context is created. This context stores the identity and can create proofs of identity. Second, the authentication context is used to do authentication with all involved entities. The important questions now become how we can control access to the authentication context, and how do we propagate it. Possibilities include the following variations:

- Access to the authentication context is made available to all trusted components.
- The caller can delegate the authentication context to the called component.
- All processes started by the user that performed the authentication inherit access to the authentication context.

The propagation of authentication context introduces a certain overhead. Therefore, we can set up so-called trusted environments in which the components can communicate without authentication. Such environments are called *protection domains.*

In J2EE, the container provides the boundary on which the authentication is performed when outside callers try to access the components inside the container. However, the container boundaries aren't always the same as the protection domain boundaries. Often the Web component container and the EJB container form a protection domain in which the EJB container can trust the Web component container (see Figure 4).

The authentication in J2EE is based on users, realms, and groups. *Users* represent people who use the information system. A *realm* is a collection of users who are authenticated in the same way. J2EE supports two realm types: default and certificate. Default realms are used to authenticate all clients except Web clients. Web clients have to use the certificate realm. Users of the default realm can belong to a group. A *group* is a category of users who share a certain role – customers or employees, for example.

For authentication JAAS supports different models to be plugged in at runtime. This enables us to use an industry-standard authentication technology through all the integrated applications. Kerberos, the most commonly used authentication technology, enables single sign-on support for multiple applications. Kerberos is a network authentication service created by MIT. (For more information on Kerberos please see http://web.mit.edu/kerberos/www/.)

JAAS, together with JSSE and JCE, enables us to build our own EAI security infrastructure. It allows us to wrap the existing applications in the same security process, using industry standards such as Kerberos for authentication, SSL and TLS for communication channel protection, and application-independent encryption engines.

In EAI projects that involve integration of non-Java applications and resources, J2EE-based security may not always be sufficient to achieve end-to-end security. A solution is to use the CORBA security model, which can be used from Java as well as from applications written in other programming languages. It's beyond the scope of this article to go into the details of CORBA security, but more information can be found at www.omg.org.

Another possibility is to use the Generic Security Services (GSS) API, developed by the Internet Engineering Task Force. GSS provides generic authentication and a secure messaging interface that supports pluggable security mechanisms. GSS version 2 is defined in a language-independent format. GSS applications can use different security mechanisms without having to make changes to the application itself.

Java bindings for GSS are referred to as JGSS. The JGSS API is part of J2SE 1.4. JGSS allows Java developers to create uniform access to security services over different mechanisms, including Kerberos.

You may wonder why we need the JGSS API. While it shares many features with JAAS and JSSE, JGSS has some features not present in the Java security model:

- The JGSS API contains support for Kerberos as the key authentication mechanism, which allows single sign-on support. The JSSE API doesn't support a Kerberos-based cipher suite – it supports SSL/TLS only.
- JGSS is a token-based API that relies on the application to handle communication. This allows an application to use the transport protocol of its choice to transport the tokens. JSSE allows only applications to use sockets.

Oracle Corporation

oracle.com / javacode

Infragistics, Inc.

www.infragistics.com

Infragistics, Inc.

www.infragistics.com

"EAI is crucial for providing online, low-latency e-commerce solutions. Therefore, sometimes e-commerce is the major reason for EAI"

- JGSS allows the selection of encryption type, which allows applications to intersperse plaintext and ciphertext messages. JSSE and JAAS don't support this.

I expect that JGSS will emerge as the standard API to use for security integration in EAI.

Naming and Directory Services

With an integrated information system, it becomes important to have a place to store environment-specific information in an application-independent way. Naming and directory services are widely used by many companies for storing information on any directory object in a computing environment, including components, computers, printers, persons, and networks. These services provide operations for creating, adding, removing, searching, and modifying these directory objects. The ubiquity of naming and directory services makes the selection of these services for integration a difficult task. Examples of naming and directory services and protocols include LDAP, NIS, NDS, DNS, SLP, and CORBA Naming.

J2EE doesn't introduce a proprietary naming and directory service. Rather, it provides an abstraction interface, the Java Naming and Directory Interface (JNDI) API, that can be used to access practically any naming or directory service in use.

To provide support for pluggable implementation, JNDI also specifies, in addition to the API, the service provider interface (SPI). This interface specifies the interfaces the provider of a naming and directory product has to implement in order to be used via JNDI (see Figure 5).

More important, by using JNDI we can access and integrate naming and directory services that already exist in our information system. The prerequisite, however, is that they provide support for JNDI – which most products do.

Web Services

EAI is crucial for providing online, low-latency e-commerce solutions – sometimes e-commerce is the major reason for EAI. EAI architecture, as proposed, is already suitable for extension into the e-commerce field using Web services, which build on top of the J2EE integration architecture. To achieve sound architecture, we have to be aware that Web services differ conceptually and technologically from business logic components (such as EJBs, CORBA, and RMI). Web services focus more on the exchange of documents, while business logic components focus on operations. Document-based systems provide looser coupling and more asynchronous communication. This corresponds with technological differences. As we know, Web services are based on XML and use XML-based protocols and technologies such as SOAP, WSDL, and UDDI.

Robust Web services can be built on top of the business logic components and should be considered highly interoperable entry points. One approach is to map the component interface to the Web service directly, which usually requires developing a dedicated component (usually an EJB component) to provide a suitable document-oriented interface. This approach is supported by the leading J2EE application servers, which provide tools to convert EJBs to Web services.

The other approach is to build Web services on workflow-based models, where each Web service calls several business components and other resources to fulfill the request. This approach requires more development work and some knowledge of the corresponding APIs. As mentioned in my previous article, the corresponding Java APIs include JAXM, JAX/RPC, JAXR, and JWSDL. These APIs are still under development and aren't a standard part of J2EE 1.3. In the meantime, we have to use custom APIs provided by application server vendors.

Web services are a promising technology and you may wonder why we don't use them for the whole EAI. The fact is that Web services as they now exist have some shortcomings: currently there is no support for transactions; the security is based on SSL/TLS only; and because Web services are an XML-based technology, they require more overhead for processing.

Performance and Scalability

Also an important EAI topic is the performance and scalability of the integrated information system. With EAI we place existing systems in a different environment and use them in ways they weren't designed for. Accordingly, it's important to test existing systems for performance and scalability before we decide how to reuse them in the integration architecture. If existing applications don't provide acceptable performance initially, we can't expect that the performance of the integrated system will be acceptable. Thus, we have to choose at the outset which existing applications meet our criteria and

Sitraka

www.sitraka.com/jprobe/jdj

which don't. For those that don't, we have to find ways to improve their performance – for example, through hardware upgrades, software upgrades, tuning, or clustering.

When integrated, existing systems will also be accessed remotely. In this sense we want to minimize the number of remote invocations and the number of layers through which calls are passed. Although I've recommended building the integration architecture in tiers, it's important to select implementation techniques carefully to minimize the communications overhead.

To achieve good performance using the J2EE integration architecture, you should focus on many different elements:

- Define the performance objectives.
- Assess the performance of existing applications.
- Design a sound integration architecture.
- Start assessing the performance in the design phase.
- Implement the integration architecture using performance optimal techniques.
- Tune the integration infrastructure for performance.
- Provide an adequate hardware platform.

J2EE Application Servers for EAI

Finally, we have to mention the selection of the J2EE application servers. We have to be aware that different servers comply with different versions of the J2EE platform specification and that some servers support value-added features. Some value-added features require the use of specialist APIs. Examples include support for Web services. Most application servers that support these features don't use JAX* interfaces (such as JAX/RPC and JAXM), but provide their own APIs.

The most important value-added features that don't require us to use specialist APIs, but are instead based on declarative configurations, include:

- Support for distributed transactions and two-phase commit
- Load balancing
- Replication and failover
- Clustering
- Performance optimizations
- Connectors for integration with EIS systems (SAP R/3, PeopleSoft, etc.)
- Connectors for management systems (SNMP, CA-Unicenter, IBM Tivoli, etc.)
- Support for pluggable middleware, including JMS, JNDI, and ORB implementations, pluggable authentication modules (PAM), etc.
- Support for automatic logging
- Support for wireless clients
- Support for hot deployment of Web and business logic components

Some vendors, already recognizing the role of application servers for integration, have supplemented their J2EE application servers and focused them directly on integration. Many of them even specifically name them as such. Such integration servers provide more tools to access existing systems and often provide out-of-the-box support for technologies necessary for EAI.

Summary

J2EE provides solutions for advanced EAI topics such as transaction, security, and naming and directory integration. The EAI architecture also can support Web services, and is actually the prerequisite for effective, low-latency e-commerce solutions.

Resource

- Juric, M.B., with Basha, S.J., Leander, R., and Nagappan, R. (2001). *Professional J2EE EAI*. Wrox Press.

AUTHOR BIOS

Matjaz B. Juric is the author of Professional J2EE EAI and coauthor of Professional EJB. He holds a PhD in computer and information science.

Ivan Rozman is the faculty dean at the University of Maribor and has many years of experience in EAI projects. He holds a PhD in computer and information science.

matjaz.juric@uni-mb.si

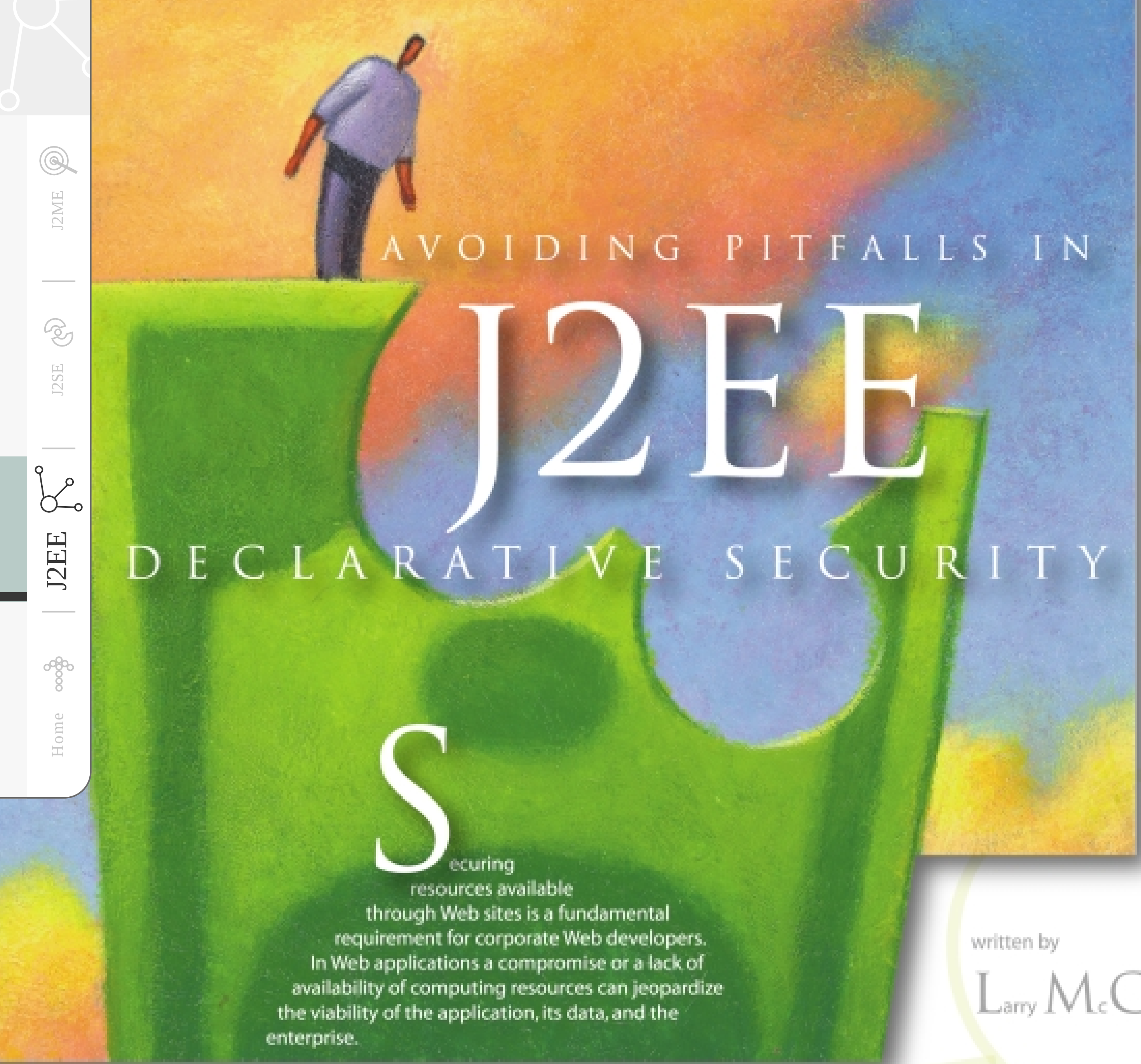
i.rozman@uni-mb.si

INT, Inc

www.int.com

Canoo
Engineering AG

www.canoo.com/ulc/



AVOIDING PITFALLS IN J2EE DECLARATIVE SECURITY

Securing resources available through Web sites is a fundamental requirement for corporate Web developers. In Web applications a compromise or a lack of availability of computing resources can jeopardize the viability of the application, its data, and the enterprise.

written by Larry McCay

Applications need to prevent improper access to data and ensure that data integrity can be maintained. To guard against such compromises, developers must first have an awareness of resource sensitivity. Proper safeguards need to be in place within the workplace and development process to ensure that sensitive information isn't inadvertently made available for improper viewing and/or modification.

This article describes certain aspects of J2EE declarative security, in particular how security constraints are used within the Web tier to protect access to resources with varying levels of sensitivity. It also illustrates the use of a sample application as a tool for determining the behavior of a given application server's constraint matching. I'll also share some of the issues I encountered while developing and deploying the sample application across several application server platforms.

While the J2EE platform strives to provide an interoperable platform for developing and deploying secure Web applications, ambiguities in the early specifications have led to important implementation differences across vendors. I'll explore some of the effects these differences have on the development of secure, interoperable applications.

J2EE Security

Authentication is the act of determining the identity of a user based on the supplied credentials. Determining whether a user should be granted access to protected resources is called *authorization*.

J2EE uses *lazy authentication* – users aren't challenged to offer proof of identity until they request access to a protected resource.

Authentication may be performed by an application using one of the following mechanisms:

- ***HTTP basic authentication:*** User's credentials collected by a Web browser
- ***HTTP digest authentication:*** Optionally supported by J2EE vendors
- ***HTTP client authentication:*** Uses digital certificates
- ***HTTP form-based authentication:*** Uses an application form to collect credentials

SilverStream Software

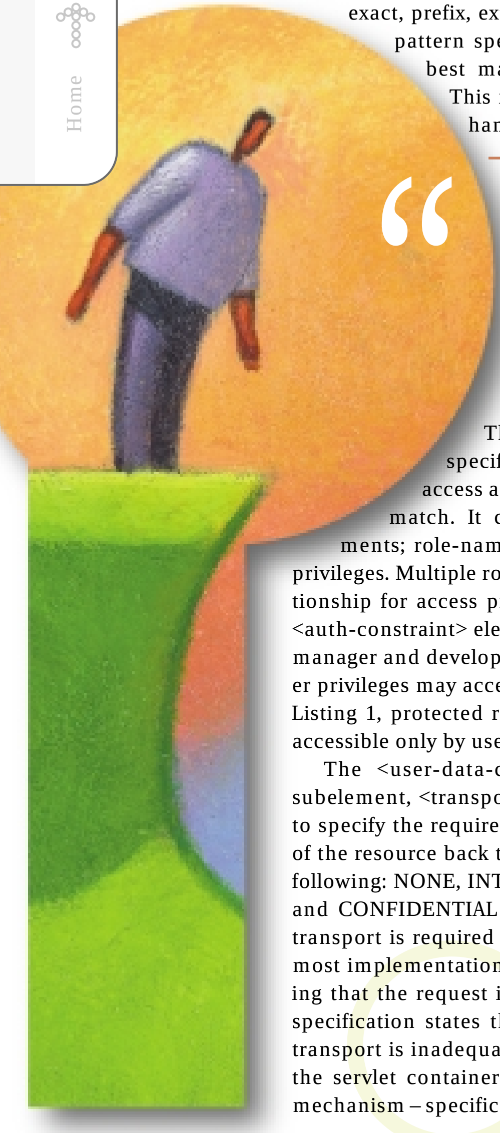
www.silverstream.com/challenge

The J2EE platform prescribes the use of a declarative language within the deployment descriptors of the components to implement container-managed security, also known as *declarative* security. Along with declarative security, the servlet and EJB specifications require a set of APIs for implementing programmatic security.

The language used to declare security policy within Web applications is composed of a set of XML elements within the deployment descriptor web.xml. A typical example of the fundamental element, <security-constraint>, is given in Listing 1. (Listings 1-3 can be downloaded from www.sys-con/java/sourcec.cfm.) The anatomy of a <security-constraint> declaration consists of a number of subelements that define the implementation of the declared piece of policy.

The Web resource element identifies the criteria from the resource request to be used in determining a match for the resource. In the case of the constraint in Listing 1, a resource, prefix-get-manager, can be identified by a request for any resource that begins with the given URL pattern, in this case a directory structure within the servletPath – in other words, anything within the directory structure /acme/widget relative to the root directory of the application.

The http-method element specifies which type of HTTP request is constrained by this declaration. In this case GET requests are constrained. In short, all HTTP GET requests for resources in the /acme/widget directory of this application are considered a match for this constraint. Four types of URL patterns may be used in the declaration of a security constraint: exact, prefix, extension, and universal. The type of URL pattern specified is important in determining the best match for a particular resource request. This is covered later in the discussion about handling overlapping constraints.



"IT'S VITALLY IMPORTANT TO UNDERSTAND THAT MISTAKES IN THE DECLARATION OF POLICY IN THE DEPLOYMENT DESCRIPTORS WILL LEAD TO PROTECTED RESOURCES BEING INAPPROPRIATELY AVAILABLE"

The <auth-constraint> element is used to specify the logical set of privileges required to access any resource for which this constraint is a match. It contains zero or more role-name elements; role-names are used to represent logical sets of privileges. Multiple role-name elements designate an OR relationship for access privileges. In other words, if a particular <auth-constraint> element contains two role-name elements, manager and developer, then users with manager *or* developer privileges may access the protected resource. In the case of Listing 1, protected resources that match this constraint are accessible only by users with manager privileges.

The <user-data-constraint> element consists of one subelement, <transport-guarantee>. This subelement is used to specify the requirements for the transport or transmission of the resource back to the client. Its value may be one of the following: NONE, INTEGRAL, or CONFIDENTIAL. INTEGRAL and CONFIDENTIAL each declare that a secure method of transport is required for access to the resource. At this time, most implementations implement this declaration by asserting that the request is being made over SSL. The Servlet 2.3 specification states that upon determining that the current transport is inadequate for accessing the requested resource, the servlet container must redirect the request to a secure mechanism – specifically SSL. At the time of this writing, some vendors implement this requirement; others simply deny access to the resource. Either approach protects the resource from inadvertent compromise.

Overlapping Constraints

Earlier I mentioned the four types of URL patterns that can be used to match a resource request. The fact is, these URL patterns could result in overlapping constraints – meaning there could be more than one match. This is where it's important to emphasize that the specified algorithm is one of *best match*, that is, the algorithm must determine which of the set of constraints matching a particular resource request is the most precise. Differences across application server platforms can be found in this area. Being unaware of your vendor's handling of overlapping constraints puts your sensitive resources at risk.

It's extremely important to understand the order of precedence for evaluating URL patterns. The algorithm is intended to be implemented such that (in descending order of precedence):

- Exact matches take precedence over prefix and extension matches.
- Prefix matches take precedence over shorter prefix matches and extension matches.
- Extension matches take precedence over universal matches.
- Universal matches, specified with a URL pattern of "/".

Using this description of the algorithm, we can define four constraints that would overlap for certain resource requests and describe the expected behavior. Given the constraint declarations in Listing 2, we can trace the matching behavior for a given set of resource requests.

First, let's describe the policy that results from the constraint declarations. All HTTP GET requests for resources that end with an .html extension require developer privileges, except for those .html files within the /acme/widget and the /acme directories. The /acme/widget directory requires manager privileges and the /acme directory allows access only to administrators and junior administrators. The /acme/widget/admin.html file is an exception in that it explicitly requires a role of administrator.

HTTP GET requests for resource access to /index.html match the extension-mapping constraint and access is constrained to users with developer privileges.

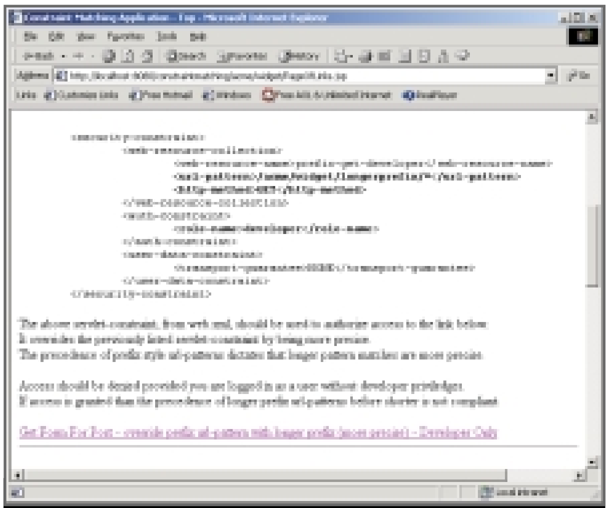
HTTP GET requests for resource access to /acme/admin.html match the prefix-mapping constraint for the /acme directory and access is granted to users with junior_administrator or administrator privileges.

HTTP GET requests for resource access to /acme/widget/index.html match the prefix-mapping constraint for the /acme/widget directory and access is granted only to users with manager privileges. This match is best because the URL pattern match is longer and therefore more precise than /acme/*.

HTTP GET requests for resource access to /acme/widget/admin.html exactly match the mapping constraint for the /acme/widget/admin.html file and access is granted only to users with administrator privileges. This match is best because

SpiritSoft

www.spiritsoft.com/climber



```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>prefix-get-developer</web-resource-name>
        <url-pattern>/acme/widget/longerprefix/*</url-pattern>
        <http-method>GET</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>developer</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

The above servlet-constraint, from web.xml, should be used to authorize access to the link below.
It overrides the previously listed servlet-constraint by being more precise.
The precedence of prefix style url-patterns dictates that longer pattern matches are more precise.

Access should be denied provided you are logged in as a user without developer privileges.
If access is granted then the precedence of longer prefix url-patterns before shorter is not compliant.

Get Form For Post - override prefix url-pattern with longer prefix (more precise) - Developer Only

FIGURE 1 Constraint-matching application

it's an exact match between the requested servlet path and the URL pattern, and is therefore more precise than the extension and prefix mappings.

Differences Across Vendors

I mentioned previously some ambiguities in versions of the Servlet specification prior to 2.3 that have led to multiple interpretations of the constraint-matching algorithm and therefore differences in implementation across vendors. It's possible to encounter implementations that are closer to a *first-match* algorithm than to a best-match. In these cases the search for a match ends immediately on finding a match; that is, there's no attempt to find the most precise match across a set of overlapping constraints. If we look at the previous example, a request for access to /acme/widget/admin.html potentially would grant access to a junior administrator. This is clearly not what is intended by the application developer, as seen by looking at the deployment descriptor.

The provided sample application aids in determining the implementation behavior for any given J2EE application server platform. I'll discuss the development and use of this application shortly.

Compromise Through Configuration

In addition to vendor algorithm differences, it's vitally important to understand that mistakes in the declaration of policy in the deployment descriptors will lead to protected resources being inappropriately available. For instance, developers who want to protect all spreadsheets within an application may be inclined to use an extension-type URL pattern. While this pattern declares that all resources with extensions of type .xls are constrained to users with manager privileges, a prefix constraint for a directory containing spreadsheets has a less restrictive constraint. Listing 3 shows the constraint declarations for this example, which illustrates the importance of understanding the matching algorithm as well as the existing policy language within an application when adding resources to be served by the application. Without realizing that spreadsheets are protected within the application by an extension-type constraint, a developer may add a spreadsheet file to a directory with less restrictive constraints.

Sample Application Goals

The sample was initially intended as an application that would illustrate the behavior of the constraint-matching algorithm. It became clear early on in the testing of the sample that there were differences between application server platforms and that the application could be used to help discern how a given platform would behave.

The application consists of a set of JSP pages that are declared as being of varying levels of sensitivity.

The sample is designed so that it renders the actual constraint that should match to each link within the pages and the expected behavior when the user is logged in with manager privileges and not developer privileges (see Figure 1).

Deployment of Sample Application

To deploy the sample application to your environment:

1. Download the sample application, in the form of a standard WAR file, from www.sys-con.com/java/sourcec.cfm.
2. Deploy the application according to your application server's deployment instructions. If you need to acquire an application server to run the sample, see the resources section at the end of the article.
3. Have a user with manager (not developer) privileges available.

Running the Sample Application

To run the sample:

1. Start the application server.
2. Initiate a request for the default welcome file with http://localhost:9090/constraintmatching, where the port is determined by the application server (HP-AS uses 9090 and Tomcat uses 8080 out of the box).
3. Log in as the user with manager privileges.
4. Use the links on the subsequent pages, observing the behavior and comparing it with the expected outcome described in the pages.

Lessons Learned

I learned a number of lessons with regard to interoperability during the development of the sample application. Since the application was developed using HP-AS 8.0, the sample deployed and ran flawlessly on this platform. It was during the testing of the application on Jakarta Tomcat and BEA WebLogic Server 6.1 that I realized changes were needed to maximize interoperability.

The most significant changes were required for deploying the application to Apache's Jakarta Tomcat environment. Given the application's initial deployment descriptor, it was obvious that the algorithm wasn't implemented as expected. A post to the Tomcat developer mailing list quickly resolved the issues.

The implementation, as expected, follows the guidelines outlined by the Servlet 2.3 expert group. Though the specification clearly reads as a best-match algorithm, the expert group agreed that the Tomcat and J2EE 1.2 and 1.3 beta reference implementations' constraint processing, being first-match, was simple and sufficient.

Tomcat's implementation of the constraint-matching algorithm requires that the constraints appear within the deployment descriptor in the order of precedence required for the different types of URL patterns, including the need to sort prefix-type constraints by URL pattern length. This is *not* an intuitive requirement; I'm more inclined to logically group constraints based on the URL pattern or the <auth-constraint> so as to make the policy more readable. Tomcat's implementation is a first-match algorithm rather than a best-match, and this is why the constraints need to be so ordered. It seems to me that this implementation is more error-prone than implementing the algorithm directly in the container. It *does* give the well-informed developer the ability to tweak the algorithm by ordering the constraints in the desired order of precedence; however, it renders the access policy nonportable.

AUTHOR BIO
Larry McCay, a senior architect for Hewlett-Packard, has been designing and developing software for over 10 years. Larry has been a major contributor to the Hewlett-Packard Application Server and Netaction Platform and is active within the JCP in the area of security.

Compuware Corp.

www.compuware.com/products/optimalj

For deployment on BEA WebLogic Server 6.1, there was a bit of annoyance in that WLS uses the realm-name element within the login-config to determine the WLS realm implementation for authentication. Once that was resolved and changed in the deployment descriptor, the application ran fine. One interesting behavior seen in WLS and not HP-AS or Tomcat was that when an authenticated user has inadequate privileges to access a resource, the user is asked to reauthenticate. The other platforms tested simply deny access.

Another difference that was uncovered results from having multiple constraints of equal type match a given request. The Servlet 2.3 specification isn't clear on how to handle this situation. The deployment descriptor of the sample application has two constraints that are identical except in the <transport-guarantee> declaration. On HP-AS and Tomcat the first of the two best-matching constraints is selected due to the algorithm implementation for each of the application servers. WLS apparently maps the constraint internally such that identical constraints, in terms of URL pattern and http-method, overwrite each other. This leads to selection of the last-matching constraint. Currently, this type of configuration should be considered programmer error; as a result, since it isn't clear what an application server should do in this event, resources may be accessible differently across application servers. The specification should specify that ambiguous constraints – even when a result of programmer error – be handled a particular way. This is likely to be addressed in an effort currently under way under the Java Community Process (JCP).

“DUE TO THE DIFFERENCES ACROSS VENDORS, THE DEVELOPMENT OF PORTABLE APPLICATIONS REQUIRES SPECIFIC WAYS OF DECLARING POLICY AND STRUCTURING THE DIRECTORIES WITHIN AN APPLICATION”

The deployment descriptor for the sample was changed to specify “filerealm\” as the realm-name, because this is the default realm for WLS out of the box, and HP-AS and Tomcat don't require anything in particular. In addition, the two prefix-type constraints that match the same request have been ordered to align with the behavior of HP-AS and Tomcat when processing multiple constraints for a given request. To make the deployment descriptor more portable, these constraints should be consolidated into one. I've left that as an exercise for the reader.

For additional fun with the sample application, change the order of the constraints in the web.xml file and observe behavior changes – especially within Tomcat.

Development Ramifications

The most important lesson learned is that, due to the differences across vendors, the development of portable applications requires specific ways of declaring policy and structuring the directories within an application.

Developers should provide resources within a flatter directory structure and place the constraints in the deployment descriptor in the order of precedence instead of having directories and subdirectories with varying levels of sensitivity, break these into separate directory structures off the application root. A directory structure such as /public/manager/administrator, for example, should be broken into three separate directories, for example, /public, /manager, and /administrator.

Ordering of constraints within the deployment descriptor should be as follows:

- Exact matches
- Prefix matches
- Extension matches
- Finally, universal match

Also, due to the way that multiple matches for the same URL patterns and HTTP methods are handled across vendors, developers should consider the declaration of multiple constraints of the same type for a given URL pattern/http-method combination as an error in declaration. That is, until there are specific tests for these cases in the Technology Compatibility Kits (TCK) for the Servlet/J2EE specifications. J2EE 1.4 should contain rigorous tests in all these areas. Until then, developers need to ensure that there is only one constraint that will be considered the best match for any given request.

As development tools for J2EE application development, assembly, and deployment mature, much of the direct editing of the deployment descriptors is eliminated – possibly making it more difficult to police these issues. Be aware of how your tool orders the constraints within the deployment descriptor so as to ensure policy portability across vendors.

A Brighter Future

An effort is under way within the JCP to formalize a contract between container vendors and authorization service providers. While this effort is focused primarily on the integration of external security providers, J2SE and J2EE, and application servers with enterprise policy infrastructure, there is an important side effect that will aid in the area of constraint processing. In order to license the technology of this effort, authorization providers and applications server vendors alike will need to pass the TCK that corresponds to the SPI contract. This TCK will also be included in the J2EE 1.4 TCK and provide rigorous tests of security constraint processing. Therefore, J2EE 1.4 will offer a more portable environment for J2EE declarative security.

Conclusion

The realization of a secure, interoperable Web application development platform is within reach. In the meantime, care must be taken in defining policy through the declarative language built into the J2EE platform so that portability across vendors will be maintained. In addition, careful evaluation of an application's declared policy, along with knowledge of the vendor's algorithm implementation, is required to deploy existing applications with protected resources to new application server platforms.

Resources

- *HP-AS 8.0 – the free J2EE Application Server:* www.hpmiddleware.com
- *Apache Jakarta Tomcat:* http://jakarta.apache.org/tomcat
- *BEA WebLogic Server:* www.bea.com
- *Servlet 2.3 Specification:* http://jcp.org/aboutJava/communityprocess/first/jsr053/index.html

lawrence_mccay-iii@hp.com

Precise Software

www.precise.com/jdj

Protecting Commercial JSP Applications

Take advantage of the benefits of JSP

WRITTEN BY
MICAH SILVERMAN

A major roadblock to using any of the server-side scripting architectures for developing commercial software is the fact that (traditionally) the source code must be delivered to customers when deploying applications.

Java source code is compiled into an intermediate code called *bytecode*, and the Java Virtual Machine (JVM) interprets this bytecode directly. It's the bytecode that makes Java class files completely platform-independent. Not only is the bytecode easy to decompile, but the descriptive variable names are included in it (and thus in the decompiled source code), making it much easier to understand the decompiled source code. This presents another formidable roadblock to deploying commercial Java-based software.

This article outlines a technique to protect JSP-based applications in such a way that they can be deployed to customers without giving away source code or class files that are easy to decompile. This technique employs features of the Java 2 Platform, Enterprise Edition (J2EE) Web application specification and a bytecode protection technology called *obfuscation*. A detailed example is provided that enables you to better understand the issues and the solution.

JavaServer Pages (JSP) provide a rapid development and deployment analog to Active Server Pages (ASP) with a few significant advantages. Servlet source code is generated from the .jsp files in the form of .java files. These are then compiled into standard servlet .class files.

These servlet classes are loaded into a server (referred to as a container in Java nomenclature). The container routes JSP requests to the corresponding class. With ASP, the source code is actively interpreted at the server and the response is sent back to the client. With JSP, the Java bytecode is preloaded into the container, making responses to requests highly efficient.

Web Application Architecture

The Web application specification (http://java.sun.com/products/servlet/2.2/index.html, section 9) allows JSP applications to run on any platform and in any vendor's J2EE-compliant container. It specifies a standard directory structure to hold static content (e.g., HTML pages and images), JSPs, servlets, and supporting Java classes. In addition, it defines a deployment descriptor – an XML file that conforms to a document type definition (DTD) found at http://java.sun.com/j2ee/dtds/web-app_2_2.dtd. The deployment descriptor defines metainformation about the application to the container. This can include global variables called *context parameters*, servlet definitions and their initialization parameters, and URL mappings.

Since generated classes from JSP files are servlets, we'll see that using servlet definitions and URL mappings enables us to deliver a JSP application without the source code found in the .jsp files.

The following is an example of a Web application directory structure. Required elements are in bold face:

```
MyApp/
 index.html
 processFunctions.jsp
 images/
  logo.gif
 WEB-INF/
  web.xml
  classes/
   HTMLUtils/
 taglib/
```

Note: The taglib and classes directories are required only if the application uses tag libraries (see http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPTags.html for more information) or supporting classes, respectively.

JSP Life Cycle

Application containers that support JSPs go through the following general steps during development:

1. Java source generation
2. Java source compilation
3. Resultant servlet class installation and content delivery

Changes made to a JSP file during development trigger an automatic sweep through the steps listed above, which helps in the ability to rapidly develop JSP. However, there's a certain amount of overhead, including a file-system check to determine if a file has changed for every request. Therefore, when a JSP (or Web application) is ready to be deployed, it's better to turn off the dynamic generation/recompilation features.

It's important to note that the code running in the container that's servicing requests is the compiled .class file(s), not the .jsp file. This is the first step in eliminating the need to distribute the .jsp files. The other steps involve configuration issues outlined in more detail in the example Web application discussed later. For more information on the life cycle of a JSP, go to http://java.sun.com/j2ee/tutorial/1_3fcs/doc/JSPIntro4.html.

Obfuscation

Obfuscation tools have gone through at least two generations to date. The first generation analyzed a group of class files and replaced class, method, and variable names with meaningless ideneXcelon Corporation

www.exln.com

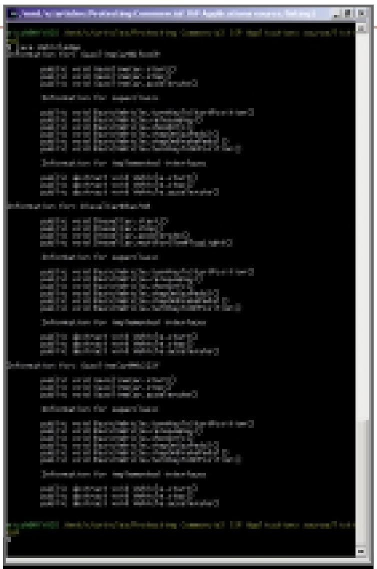


FIGURE 1 Regular version's output

FIGURE 2 Obfuscated version's output

tifiers. As the obfuscator parsed the files, it would internally keep track of mappings from the original identifiers to the new (meaningless) identifiers. This made the source code obtained from decompilation much more difficult to analyze and understand.

The code in Listing 1 is a small Java application that uses typically descriptive class and method names; the code in Listing 2 is an obfuscated version. (Listings 1–9 and additional source code can be downloaded from the ***JDJ*** Web site at www.sys-con.com/java/sourcec.cfm.) It's much more difficult to glean functional information from the decompiled obfuscated code. An end user would run both versions of the application the same way (java VehicleApp). Note that the method BasicVehicle .showInfo() uses Java's reflection classes to avoid a very important shortcoming in first generation obfuscation tools: string literals are not obfuscated in any way. String literals often give away information about what is happening in the code.

Figures 1 and 2 show the screen output from the run of the regular and obfuscated versions, respectively.

Second-generation obfuscation tools often provide a facility to encrypt string literals and will also rearrange certain code blocks, such as loops, to be more confusing.

An interesting side effect of obfuscation is that the resultant class files are usually smaller due to the shortening of class, method, and field names. Some second-generation obfuscators even claim a performance improvement because of the way the code is rearranged.

Many obfuscation tools offer the option of using "illegal" identifiers. These identifiers don't conform to the bytecode specification in the Java Language Specification. Most (current) JVMs will still work with these identifiers while many decompilers won't. The perceived advantage in foiling decompilers is not worth the risk that the code you deliver won't work when a client updates his or her JVM at some point. This feature should always be disabled. It's important to note that obfuscation, even the kind that encrypts string literals, normally results in perfectly legal (if confusing to humans) bytecode.

Sample Web Application

Application Overview

The example Web application is simple; it has a number of features that highlight the JSP life cycle and show how to break the dependence on the original .jsp source code files and the resultant servlet .class files. These features, listed below for reference, are described in more detail as the example is analyzed:

- Self-referential links
- Included JSP files
- Supporting class files
- Context parameters (global to the Web application)
- Error page support

Listing 3 provides all the source files that make up PopQuiz, the sample application. A user can select from a list of multiple-choice quizzes. The questions are displayed in a random order as is each answer. The application keeps track of the order of the questions and answers that a particular user receives, along with the user's answers and the correct answers. A review of the quiz, highlighting right and wrong (or unanswered) questions, and a score are given when the user submits the quiz for grading. The quizzes are organized within the Web application's directory tree as a set of XML documents. Listing 4 provides the DTD and Listing 5 the sample quizzes.

The Tomcat application server, a subproject of the Apache Jakarta project (http://jakarta.apache.org), was used to test the Web application. Note, however, that the Web application should run in any J2EE-compliant container as is. The Xerces XML library, part of the Apache XML project (http://xml.apache.org), was used for the XML parsing. The flow through the application is as follows:

- ***index.jsp:*** Shows available quizzes
- ***TakeQuiz.jsp:*** Displays quiz
- ***GradeQuiz.jsp:*** Displays quiz results and score

If any of a number of error conditions occur (bad quiz file name, exceptions on XML parsing, etc.), the JSP error page Error.jsp is displayed. Each of the above mentioned files has an include reference to common/GlobalHeaderVars.jsp. A number of global variables are defined in this file.

Self-referential links, as well as links to the other pages, illustrate an important feature in eliminating the need for the .jsp source files: URL references won't change throughout the application. This preserves an important feature, one that makes JSP technology attractive – the ability to rapidly develop and deploy.

The included common/GlobalHeaderVars.jsp file illustrates an aspect of the JSP servlet generation life cycle. Included files are integrated into the JSP and then a .java file is generated. Since the included file is never referenced directly (through links or form submissions), the original source doesn't need to be referenced in the deployment descriptor (WEB-INF/web.xml, see later).

Supporting class files contained in the WEB-INF/classes directory are automatically included in the container's classpath. No special referencing or classpath manipulation is required when delivering a Web application in general, or in the case of the special structure we're examining here.

Context parameters found in the deployment descriptor (WEB-INF/web .xml) are utilized in the same way whether the page is a JSP or a servlet. Error pages are a convenient mechanism for forwarding to a JSP in the event of an exception. Ordinarily, the container would be responsible for displaying information when an exception occurs. The content of this information varies from container to container and usually shows a stack trace, which is not very

Altova

www.altova.com

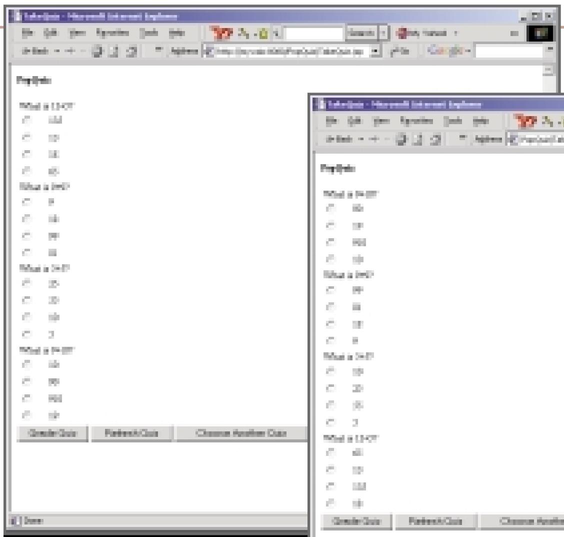


FIGURE 3 Original Quiz

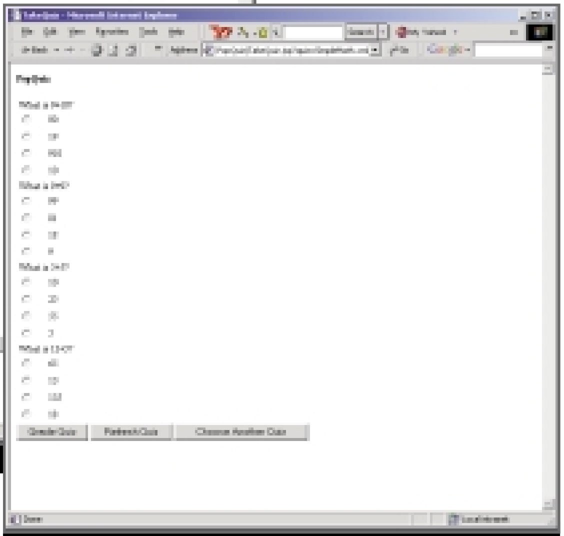


FIGURE 4 Refreshed Quiz

useful to the average user. The error-page mechanism allows for a customizable error page that can have the same look and feel as the rest of the application. In the sample application, the error page displays a single meaningful exception string along with links to get back into the application.

Listing 6 provides the original deployment descriptor; Listing 7 shows the modified deployment descriptor. Figures 3 and 4 show screenshots of the same quiz in a different order. The "Refresh Quiz" button can be clicked to show how the application automatically scrambles questions and answers.

Next Steps

We now need to generate and compile the .java files. The resultant class files will be referenced from a modified deployment descriptor. At that point, the original .jsp files will no longer be needed. If the class files were obfuscated as outlined above, it would provide a reasonably safe way to deliver a Web application commercially. *Safe* in this context refers to protecting the original source code from reverse-engineering. It should be noted that without obfuscation, the class files aren't much more protected than the original source files.

JSP to Java Source Generation

Ordinarily, the container is responsible for generating .java files from the .jsp source files, compiling the .java source files, loading the resultant class files into the container, and delivering the content to the client. By visiting each page in the application, we could rely on the container to generate the class files and then reference these class files in the deployment descriptor. This has two drawbacks: the generated class files typically have cryptic or long names (here's a Tomcat example:_0002fTakeQuiz_0002ejspTakeQuiz_jsp_0.java), and development and deployment become increasingly difficult to maintain this way.

Most containers provide command-line or GUI utilities for manually generating the .java files from the representative .jsp files. Usually this is simply a command-line interface to the same classes that generate the .java files on the fly. These utilities provide a more manageable way to generate the entire application at once. Convenient switches are also provided to allow the entire application to be generated in a particular Java package. Assuming the following directory structure:

```
PopQuiz/
 Error.jsp
 GradeQuiz.jsp
 index.jsp
 TakeQuiz.jsp
 common/
  GlobalHeaderVars.jsp
 quizzes/
  Geography.xml
  PopQuiz.dtd
  SimpleMath.xml
 WEB-INF/
  web.xml
  classes/
   com/
    MPowerIT/
      io/
       XMLFileFilter.class
     quiz/
       Question.class
   Quiz.class
```

The following Tomcat command (run from TOMCAT_HOME):

```
 jspc -p com.MPowerIT.servlet -d
webapps\PopQuiz\WEB-INF\classes -
webapp webapps\PopQuiz
```

will generate the .java files in the com.MPowerIT package and place the resulting .java files within the Web application's classes directory. Remember that anything under WEB-INF/classes is automatically included in the classpath. Now, we can compile the .java files (and obfuscate the class files), remove the .jsps, and edit the web.xml deployment descriptor file to properly reference the class files. Note that the file GlobalHeaderVars.java will also be generated but isn't needed as it's incorporated into the other files. It can be safely removed. Here's the resultant directory structure (with the .jsp files removed):

```
PopQuiz/
 quizzes/
  Geography.xml
  PopQuiz.dtd
  SimpleMath.xml
 WEB-INF/
  web.xml
  classes/
   com/
    MPowerIT/
     io/
        XMLFileFilter.class
     quiz/
   Question.class
   Quiz.class
  servlet/
   Error.class
   GradeQuiz.class
   TakeQuiz.class
   index.class
```

The Deployment Descriptor

The final and crucial step is editing the web.xml deployment descriptor. Listing 7 contains the complete contents of the file. The <servlet> and <servlet-mapping> sections allow us to control how the application is accessed:

```
…
<servlet>
 <servlet-name>index</servlet-name>
 <servlet-
class>com.MPowerIT.servlet.index</serv
let-class>
</servlet>
…
<servlet-mapping>
 <servlet-name>index</servlet-name>
 <url-pattern>index.jsp</url-pattern>
</servlet-mapping>
…
```

AltoWeb

www.altoweb.com

IBM

ibm.com/websphere/ibmtools

The <servlet> tag associates a name with the servlet and references a class. The <servlet-name> tag has nothing to do with how the servlet is accessed and is strictly for use within the deployment descriptor. The <servlet-mapping> tag defines how the servlet will be accessed. The <url-pattern> tag is the key to keeping all our self-referential and internal links consistent. It's this tag that allows a URL like www.MPowerIT.com/Pop-Quiz/index.jsp to be valid even though there's no longer an index.jsp file.

Testing the Application

The original application is contained in the file PopQuiz.zip, which includes the .jsp source files. Another version of the application is available in the file CommercialPopQuiz.zip, which includes the .java source of the generated JSP. The obfuscation of the class files is left as an exercise for the reader. To be as effective as possible, however, remember to use an obfuscator that not only rewrites all class, method, and field names, but also encrypts string literals. See Listing 8 for an example of a decompiled GradeQuiz.class file. Listing 9 shows the decompilation of this same file after it's been obfuscated. The obfuscator KlassMaster (www.zelix.com) was used to obfuscate the class file; it encrypts string literals.

The application uses the Xerces DOM XML parser found in the Apache Foundation's XML project, http://xml.apache.org. The xerces.jar file needs to be in the container's classpath. This Java archive (JAR) comes with Tomcat in its <TOMCAT_HOME>/lib directory. All .jar files in the lib directory are automatically included in Tomcat's classpath, so no additional configuration is required when using Tomcat.

Assuming that your classpath is set properly as described above, you should be able to expand the .zip archives in the appropriate place for your container and run the sample application. For Tomcat, you would expand the archives in <TOMCAT_HOME>/web-apps. The URLs to access the Web applications are http://<tomcathost>:<tomcat-port>/PopQuiz/index.jsp and http://<tomcat-host>:<tomcat-port>/CommercialPop-Quiz/index.jsp. The output you get in your browser should be exactly the same in either case.

Summary

JSP technology provides for rapid development and deployment as well as efficient delivery of content. Ordinarily, the cost of this is inclusion of the source code, in the form of .jsp files, with the application. By using the deployment descriptor's features, obfuscation, and the fact that compiled JSPs are servlets, developers can take advantage of the benefits of JSP without the commercial downside.

A number of advanced JSP technologies such as the use of beans and custom tags were not covered in this article. The outlined techniques for protecting a JSP-based application apply to these other facets of the technology as well.

Resources

- *Servlet Specification:* http://java.sun.com/products/servlet/2.2/index.html
- *web.xml DTD:* http://java.sun.com/j2ee/dtds/webapp_2_2.dtd
- *Custom Tags in JSP:* http://java.sun.com/j2ee/tutorial/1_3fcs/doc/JSPTags.html
- *JSP Life Cycle:* http://java.sun.com/j2ee/tutorial/1_3fcs/doc/JSPIntro4.html
- *Apache Jakarta Project:* http://jakarta.apache.org
- *KlassMaster Obfuscator:* www.zelix.com
- *Apache XML Project:* http://xml.apache.org

AUTHOR BIO
Micah Silverman has been working in software development and computer security since the 1980s. He's been developing Java applications since the language was released in 1995. Micah is a Sun Certified Java Programmer and an ISC2 CISSP (Certified Information Systems Security Professional).

mps@MPowerIT.com

J2ME
J2SE
J2EE
Home

IBM
ibm.com/websphere/ibmtools

KEITH BROWN J2SE EDITOR

In Search of The Community

By the time you read this, JavaOne will be over and I'll have experienced my first visit to the world's grandest Java conference. For various reasons I've never been able to attend before, but this year (knock on wood), the gods have smiled on me and the constellations are positioned in my favor.

There are many reasons why I'm looking forward to San Francisco. I'm assured the sushi is divine. Can't wait to see the legendary Alcatraz, or the city in which Dirty Harry erased the scum from its mountainous streets and posed menacing and probing questions to his quarries.

But most of all, I'm looking forward to meeting *the community*. Ahh! The oft-cited Java community. You can't see it, you can't touch it, but it's talked about and frequently referenced.

How do we define this community? It's a curious beast indeed and my question is: Does the average Java developer feel a part of the community? Do you feel a part of it?

Until you visit a Java conference or perhaps a Java users group meeting, the community is virtual (except, perhaps, for your colleagues and friends). It relies on the Web, e-mail, and publications such as ***JDJ*** to be the glue that binds it together. Virtual community Web sites, modeled in a similar fashion to Hagel and Armstrong's vision of community-oriented business models (*Net Gain* by John Hagel and Arthur G. Armstrong), assist in the bonding process. Some, such as http://java.isavvix.com, are well worth a visit for forums, neat code samples, and happenings. Despite this, it can be a lonely old life coding away, perhaps in a small team, not actually meeting face to face with other Java programmers.

However, the community is not just made up of programmers and coders. It's also made up of companies, software vendors, IT managers, educators, and perhaps even those who have only a vague involvement with technology but are curious about Java. Java seems to have a stronger sense of community than any other language (Sun even popularized the word with their "Java Community Process"). Perhaps this is because of the robust sense or perception that there's a battle going on against an easily demonized enemy. There's a war to be won and, just as societies pull together during times of war, the Java community has been strongly united and has flourished since its inception seven years ago when there was a ready-made "baddie" that we could unite against.

I'm curious. Is the Microsoft C# community as loyal and healthy as the Java community? Perhaps it still needs time to mature. Maybe someone familiar with both can let us know (www.sys-con.com/java/). Is Java perceived as just as much of a threat to C# as C# is to Java?

I, for one, am looking forward to putting faces to the community and getting a real sense of who we are, what we do, what we sound like, and what we look like. For that reason, JavaOne will be a very interesting and, hopefully, fruitful experience.

I guess by virtue of the fact that you are reading *JDJ*, you have at least some contact with the Java community. But do you feel a part of it? Do you think it's useful to attend a conference if you've never been to one? To meet the rest of your community?

keith.brown@sys-con.com

AUTHOR BIO

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

J2SE INDEX

IBM

ibm.com/db2/outperform

JDiff—What Really Changed?

Comparing Java APIs

WRITTEN BY
MATTHEW B. DOAR

One of the most common questions Java developers ask after downloading a new version of a product is: "What really changed?" JDiff is an open source Java tool, based on Javadoc and developed by the author, that produces HTML documentation describing the precise API changes between two versions of a product.

This article uses JDiff to show what changed between J2SE 1.3 and J2SE 1.4, and describes how developers can use JDiff to document the changes between two versions of their own products as easily as running Javadoc.

What Changed Between J2SE 1.3 and J2SE 1.4?

Release notes are usually high-level descriptions of feature changes. Product reference manuals tend to be large. And it's hard to compare different product versions in Web browsers' windows. Sun's J2SE 1.4 product, with all its new features, is no exception. When you want to know exactly what changed between two versions of a file, "diff" is the familiar command-line tool for the job. When you want a precise comparison between two Java APIs, I suggest using JDiff.

Figures 1 and 2 show typical HTML documentation generated by JDiff. In this case, the J2SE 1.3.0 API and the J2SE 1.4.0 API are compared using JDiff 1.0.6. Figure 1 is a screenshot of some of the packages that were changed between versions, and Figure 2 shows the details for a particular class, java.lang.Throwable. Every change in the API is reported, from new methods and fields to changes in parameter types and which exceptions are thrown. Even the changes in the documentation for each class and method can be reported.

The best way to view a JDiff report is with a Web browser. (The report comparing J2SE 1.3 and J2SE 1.4 can be found at www.jdiff.org.)

What a JDiff Report Tells You

The HTML report generated by JDiff describes the differences between two Java APIs. The right-hand frame initially contains a summary of the packages that were removed, added, or otherwise changed in some way. There are links to other JDiff-generated pages that describe the changes for each package and class in more detail. To help recall what each package and class is used for, the first sentence of the Javadoc comment (a "documentation block") in the source code is shown to the right of each entry.

The layout of the report resembles Javadoc-generated HTML; to prevent confusion, JDiff uses a different colored background and all links from JDiff pages to Javadoc HTML pages are in a monospaced font.

Indexes

A good question whenever an API changes is: "What was removed?" This is because removed (and changed) constructors, methods, and fields will cause an application to fail, ideally at compile time. Constructors, methods, and fields that were newly added are less likely to cause an application to fail. JDiff provides indexes of which packages, classes, constructors, methods, and fields were removed, added, or changed. It also provides indexes of all the removals, additions, and changes. The indexes are all HTML links, but they're not underlined, so they're easier to read quickly.

The indexes are a particularly useful feature when JDiff is used to track changes in an API as a product is being developed. Each part of the team can see precisely what has changed between the different versions.

Links

One feature that makes JDiff-generated reports useful is the large number of HTML links in a report. Every JDiff package and class page has links to the Javadoc-generated HTML pages for the specific package or class, making it easy to refer to an API's existing documentation. The JDiff navigation bar contains links to the page for each class's package, and also to the top-level summary page. Just like Javadoc, there are also links to the previous and next package or class, and to the sections within a page. All pages have links to nonframe versions of the page for browsers (and users) that can't deal with HTML frames.

Features for Developers

Two useful features for API developers are the ability to track changes in documentation and statistics about the changes between two APIs. This information is generally less useful to customers who use the API, so the features are optional in all JDiff reports.

Documentation Changes

JDiff can track changes in the Javadoc comments in the source code that's used to produce Javadoc HTML. While such changes are rarely of great interest to customers, it's very helpful for developers to know how the description of a method or field has changed during the development of an API. Each changed constructor, method, and field has links to the old documentation, the new documentation, and the highlighted differences between the two. Figure 3 shows some of the documentation changes between J2SE 1.3 and J2SE 1.4.

Statistics

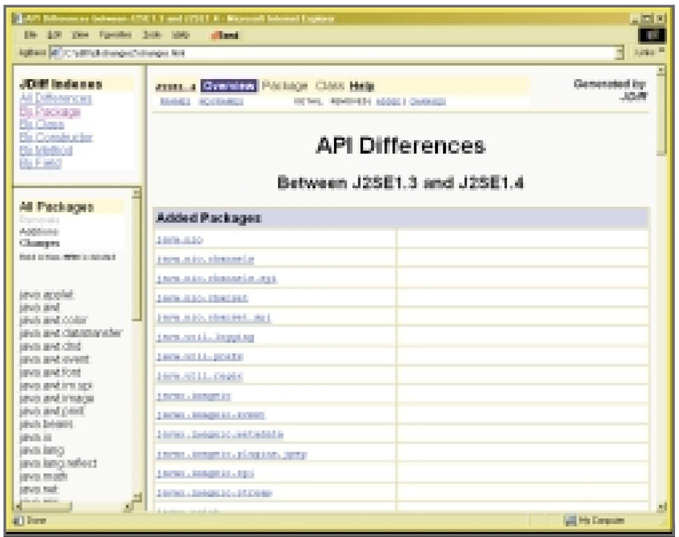
Another good question whenever an API changes is: "How much has changed?" To answer this, JDiff can track the statistics for how many constructors, methods, and fields changed in a class, how many classes changed in a package, and so on. The formula is very simple:

```
Percentage Change = 100 x (Number of
Additions + Number of Removals + (2 x
Number of Changes))/ Total Number of
Packages or Classes in Both APIs
```

For example, suppose a Java API is made up of 15 Java packages, and in the next release of the API two new packages

Mongoose Technology

www.portalstudio.com



JDiff Indexes
All Differences
By Package
By Class
By Constructor
By Method
By Field

All Packages
Removals
Additions
Changes

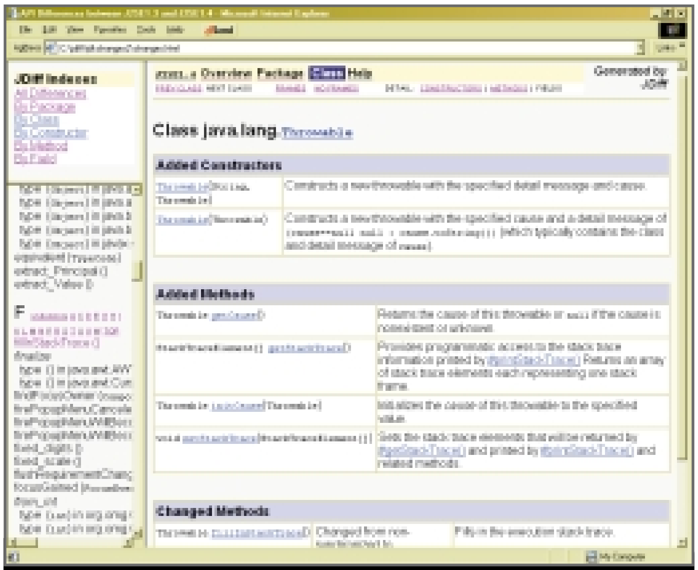
java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.im.spi
java.awt.image
java.awt.print
java.beans
java.io
java.lang
java.lang.reflect
java.math
java.net

API Differences

Between J2SE1.3 and J2SE1.4

Added Packages	
java.nio	
java.nio.channels	
java.nio.channels.spi	
java.nio.charset	
java.nio.charset.spi	
java.util.logging	
java.util.prefs	
java.util.regex	
javax.imageio	
javax.imageio.event	
javax.imageio.metadata	
javax.imageio.plugins.jpeg	
javax.imageio.spi	
javax.imageio.stream	

FIGURE 1 JDiff Report showing package changes



JDiff Indexes
All Differences
By Package
By Class
By Constructor
By Method
By Field

Class java.lang.Throwable

Added Constructors	
Throwable(String, Throwable)	Constructs a new throwable with the specified detail message and cause.
Throwable(Throwable)	Constructs a new throwable with the specified cause and a detail message of (cause==null ? null : cause.toString()) (which typically contains the class and detail message of cause).

Added Methods	
Throwable getCause()	Returns the cause of this throwable or null if the cause is nonexistent or unknown.
StackTraceElement[] getStackTrace()	Provides programmatic access to the stack trace information printed by printStackTrace(). Returns an array of stack trace elements each representing one stack frame.
Throwable initCause(Throwable)	Initializes the cause of this throwable to the specified value.
void setStackTrace(StackTraceElement[])	Sets the stack trace elements that will be returned by getStackTrace() and printed by printStackTrace() and related methods.

Changed Methods		
Throwable fillInStackTrace()	Changed from non-synchronized to	Fills in the execution stack trace.

FIGURE 2 The changes in the class java.lang.Throwable

are added and one existing package is removed so that there are now 16 packages total. Also suppose that 3 of the 16 existing packages have been changed. In this example, the percentage change between the two APIs is 100 x (2 + 1 + (2 x 3))/(15 + 16) = 29%. Using this formula, if two APIs are identical, the percentage change will be 0%, and if they're totally different, the percentage change between them will be 100%.

JDiff reports the percentage changes in each changed class, and also in each changed package, by applying the formula recursively. The percentage changes are also shown sorted in HTML tables in the report, and also in a format suitable for importing to popular spreadsheet applications. This makes it easy to identify when testing and documenting which parts of an API have changed most between different versions.

Table 1 shows the percentage changes for some popular APIs, not including documentation changes. Interestingly, the percentage change between J2SE 1.2 and J2SE 1.3.0 was about 11%, but was about 33% between J2SE 1.3.0 and J2SE 1.4.0, confirming opinions that the changes from J2SE 1.3 to J2SE 1.4 are larger than the changes in the previous major release. The breakdown of the statistics for the core Java classes show that Sun is very careful to add or change only packages and classes, and that minor releases really do contain only bug fixes, as opposed to API changes.

How to Run JDiff on Your Own APIs

As shown in Figure 4, there are three fairly straightforward steps for using JDiff. Each step involves running Javadoc, and can be executed at the command line in a script or batch file, or added to a makefile or Ant build file as part of a build process.

Step 1: ***Use JDiff to generate an XML file that represents the old API's packages.***

```
javadoc -doclet jdiff.JDiff
 -docletpath ..\..\lib\jdiff.jar
 -apiname "SuperProduct 1.0"
 -sourcepath ..\SuperProduct1.0 <old
packages>
```

This step scans the source code of the old API. The -doclet and -docletpath options are the standard options used by Javadoc to run the JDiff doclet. The apiname option creates a unique identifier for the API, and the sourcepath option indicates where to find the Java packages that make up the old API. <old packages> lists the precise packages that are scanned, just like Javadoc.

Step 2: ***Use JDiff to generate an XML file that represents the new API's packages.***

```
javadoc -doclet jdiff.JDiff
 -docletpath ..\..\lib\jdiff.jar
 -apiname "SuperProduct 2.0"
 -sourcepath ..\SuperProduct2.0 <new
packages>
```

This step scans the source code of the new API, located in the "SuperProduct2.0" directory. The new API is given the unique identifier of "SuperProduct 2.0".

Step 3: ***Use JDiff to compare the contents of the two XML files and generate an HTML report of the differences.***

```
javadoc -doclet jdiff.JDiff
 -docletpath ..\..\lib\jdiff.jar
 -d newdocs -stats
 -oldapi "SuperProduct 1.0"
 -newapi "SuperProduct 2.0"
 -javadocold "../../olddocs/"
 -javadocnew "../../newdocs/"
..\..\lib\Null.java
```

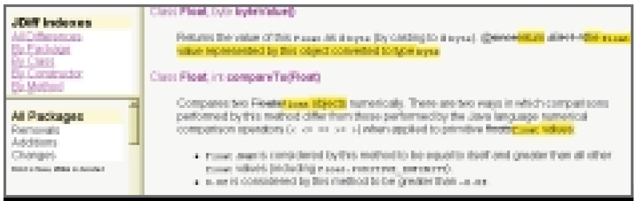
The final step compares the "SuperProduct 1.0" API and the "SuperProduct 2.0" API, with links to the Javadoc documentation in the olddocs and newdocs directories, respectively. The -d option makes the HTML report generated by JDiff appear in the directory newdocs\changes.html, and the -stats option reports statistics about the differences between the APIs. The file Null.java is present only because Javadoc has to read in at least one file, even if the doclet doesn't use it.

Interland

interland.com

OLD API	NEW API	PERCENTAGE CHANGE
J2SE 1.2	J2SE 1.3.0	11%
J2SE 1.3.0	J2SE 1.4	33%
J2SE 1.3.0	J2SE 1.3.1	<1%
EJB 1.1	EJB 2.0	37%
Servlet 2.2	Servlet 2.3	49%

TABLE 1 Percentage changes between different APIs


JDiff Indexes
All Differences
By Package
By Class
By Constructor
By Method

All Packages
Removals
Additions
Changes

Class Float, byte byteValue()

[illegible]

Class Float, int compareTo(Float)

[illegible]

FIGURE 3 Typical J2SE1.4 documentation changes

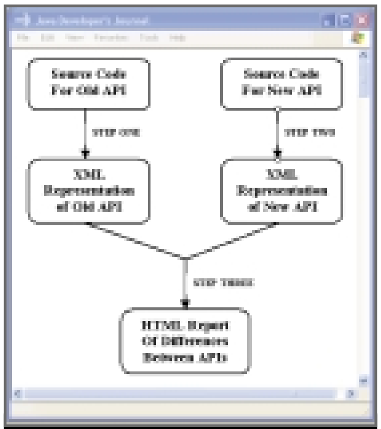

Java Developer's Journal
Source Code For Old API
Source Code For New API
STEP ONE
STEP TWO
XML Representation of Old API
XML Representation of New API
STEP THREE
HTML Report Of Differences Between APIs

FIGURE 4 Three steps for using JDiff

To make the JDiff report more useful, it helps if there's existing Javadoc HTML documentation for both APIs for HTML links from the report. It also helps if there are Javadoc comment blocks in the source code for the packages from each API, since they're used in the right-hand summary text in each entry in the report; however, you don't need them to compare APIs. You can even compare the APIs in two JAR files, where documentation blocks are not available.

JDiff also lets you write specific comments for each change between two APIs. You write a comment for each change into a "comments.xml" file with any text editor, and this file is read in when the report is generated. The comments file is regenerated after the report is finished so the comments are not lost after being incorporated into the report's HTML files. If no comments are provided, by default JDiff does its best to find appropriate comments for each change from the Javadoc comment blocks in the source code.

Scaling Issues

When scanning large APIs such as J2SE 1.4, which has 129 different packages in it, Steps 1 and 2 can take a few minutes to run on a 450MHz, 256MB Pentium III machine. Once the XML files have been generated in Steps 1 and 2 though, they don't need to be regenerated each time a JDiff report is created. Step 3 can be repeated with different options each time, using the same XML files from Steps 1 and 2. In Step 3 it took about three minutes to generate a JDiff report for the two J2SE APIs running on the same 450MHz, 256MB Pentium III machine.

With large APIs, the XML files generated in Steps 1 or 2 can be quite large – about 20MB in the case of J2SE 1.4, if all the documentation is included for comparison. These files can be archived with each release to avoid having to regenerate them later on. They can also be made smaller if changes in documentation are not tracked, since much of the content of each XML file is the entire documentation from each API's Javadoc comment block. The –firstsentence option can be used in Steps 1 and 2 to minimize the size of the XML files by storing only the first sentence of each Javadoc comment block in the XML file. The -docchanges option can be used in Step 3 to avoid tracking changes in documentation, which reduces both the size of the report and the report's index files.

How JDiff Works

JDiff uses the Javadoc doclet API (see the Javadoc homepage), which gives doclet developers a ready-made, easy-to-use tree structure of all the Java packages and classes in the files scanned by Javadoc. Doclets have been used to generate MIF and RTF documentation of APIs, to create customized Javadoc tags such as @todo, and even to generate source code for other applications using tags in the Javadoc comment blocks (see articles referenced at www.doclet.com).

The JDiff doclet has two modes of operation. In the first mode (Steps 1 and 2), it acts as an XML-generating doclet that traverses all the known packages and classes and writes as much information about them as it can into an XML file. The XML file now represents everything the Javadoc knew about the scanned API. JDiff can also generate an XML Schema file (api.xsd) that describes the XML file and permits XML parsers to validate the XML file later on.

The second mode of operation (Step 3) takes two such XML files as input to an XML parser, such as the Xerces XML parser, and carefully compares them, populating an instance of the JDiff APIDiff.java class as it does so. The results of the comparison are then used by a number of JDiff classes to generate the HTML output. A summary page of all the packages that were removed, added, or changed is created, with links to pages for each package and class. Index pages are also generated for all the differences. Finally, statistics pages and other optional HTML pages are generated. All generated files except the top-level summary are in a single directory named "changes," which makes shipping the HTML files very easy.

The Benefits of Using JDiff

- JDiff is based on the standard Javadoc tool, and is just about as simple to use.
- Developers and documentation teams can produce release documentation describing precisely what has changed in each version of their product. Knowing what has changed between versions of a product leads to faster acceptance of new versions, and fewer frustrated customer calls when a product changes.
- Developers who work in different locations and time zones can use JDiff to summarize the changes in APIs and documentation blocks as the APIs change during development.
- QA and testing organizations can use JDiff to help identify which parts of an API have changed most between versions, indicating which areas need the most testing.

Resources

- *JDiff:* www.jdiff.org,
- *Project hosted by SourceForge:* http://javadiff.sourceforge.net
- *Javadoc Tool:* http://java.sun.com/j2se/javadoc/index.html
- *Writing your own doclet:* http://java.sun.com/j2se/1.4/docs/tooldocs/javadoc/overview.html
- *Third-party doclets:* http://java.sun.com/j2se/javadoc/faq.html#doclets
- *Doclets:* www.doclet.com

doar@pobox.com

AUTHOR BIO
Matthew Doar, a software developer at Vitria Technology, Inc., has worked with Java since the early days of JDK1.1. He wrote JDiff with the intention that it should be as easy to use as Javadoc, and to improve the overall level of documentation shipped with all Java products. Matthew holds a PhD in computer science from the University of Cambridge. (www.pobox.com/~doar)

Capella University

www.capellauniversity.edu

A Multiple Application Launcher

written by
Kirk Pepperdine

Reduces the overall demand for system resources

Just recently, there was a question posted in the Java Ranch's (www.javaranch.com) Java Performance discussion group asking if there is any advantage to running more than one server application in the same VM. Group members were quick to point out that each JVM consumes a significant quantity of system resources.

By necessity, the operating system loads each VM into its own separate process slot. Consequently, each VM is forced to duplicate the other's initialization efforts and resource allocations. Specifically, each VM is required to load the JDK core classes into their respective heap spaces. Running several Java applications in the same VM would eliminate this duplication of effort. There's an analogy in the world of operating systems.

System engineers have long recognized that portions of statically linked executables contain read-only text that's potentially shareable. They knew that certain performance benefits could be realized if processes could share this text. This knowledge motivated them to develop shared-text libraries. These libraries are loaded only once and are dynamically linked to executables. Nowadays, most operating systems support the concept of a shared text library (or DLL – dynamic-link library). For example, dynamically linked UNIX processes make references to the shared library libc.so. The OS ensures that these processes share the text portion of that library.

By introducing shared libraries, system engineers reduced the overall requirements for system memory. One side effect: the scheme also places less stress on the IO channels and other system resources. This example hints at performance benefits that might be available if you could run several Java applications in the same VM.

Returning to the original thread of discussion and the question, "How do you launch multiple applications?", I suggested a way you might easily build this capability. Even though I was confident that my post was correct, I had never actually built an application launcher. A few seconds after I hit the send button I decided to stick around and read my own advice. I read my posting and thought, "Is it really that easy? Maybe I'd better check out my own advice." What follows are the lessons I learned as I "ate my own dog food."

What I'm Looking For

I'd like to have the flexibility to run several Java 2 applications concurrently in the same JVM. These applications need to function in a secure environment that isolates them from each other to the greatest extent possible. The use of this multiple application launcher (MAL) should be transparent to the application. Finally, all applications should be able to share common read-only resources. Let's start by investigating the current launch technology to see what we can learn and possibly reuse.

Launching a Java Application

MAL needs to mimic the behavior of the standard application launcher. I began my voyage of discovery by reviewing the way a Java application is loaded and then executed. I quickly discovered that starting up a VM initiates a complex sequence of events. Once the initialization sequence has completed, I'm left with a VM that's ready to load a Java class and invoke a standard entry point method. (The complete source code for this article can be downloaded from www.sys-con.com/java/sourcec.cfm.)

By default the first optional parameter found on the command line specifies the Java class that implements the entry-point method. The launcher uses the java.class.path property to find and then load this class. Reflection is used to find the entry-point method. Since it's well documented, the method needs to be declared as public static void main(String[] args). Once it's loaded and a reference to the method main has been resolved, the launcher will invoke that method. The VM will continue to function until System.exit() or Runtime.halt() is called. The launcher is the most likely candidate to make that call.

From this description, I started to see the first set of problems that needed to be addressed. The default launcher required the application's classes to be on the system's classpath. If I don't plan on shutting down the VM, I'll need to know about every application that will be run so I can configure the classpath before the VM is started. If I do introduce a new application, I'll be required to shut down the VM, reset the classpath, and then restart the VM. But the original post was about running applications in a server environment. Forcing a VM shutdown in this environment would negate the reason for using MAL.

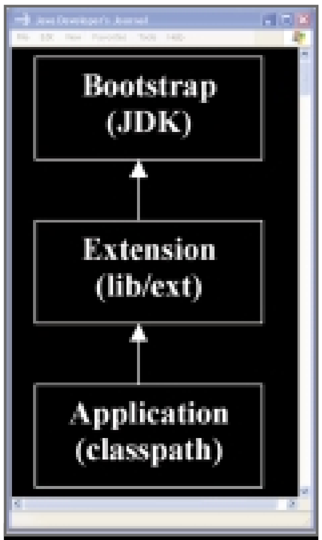


FIGURE 1 ClassLoaders

There's another related problem. The consequence of being able to use only a single classpath is that MAL can't run different versions of the same application. There seems to be no practical way to overcome this difficulty using a single classpath.

Both of these scenarios point to a new requirement. I should be able to dynamically extend the VM's classpath in a secure manner. I need a way to isolate these "on-demand" extensions of the classpath to the launching of the application in question. Since this discussion is centered on classpath and class loading, let's turn our attention to the Java 2 ClassLoader model.

Java 2 ClassLoaders (Part 1)

Java 2 introduced a new class-loading architecture, commonly known as the delegation model. At its basis is the notion that ClassLoaders can be chained. In this chain, every ClassLoader has a parent and a ClassLoader's parent will be given the first chance to load a class. The only exception to this rule is the bootstrap ClassLoader. It's implemented in the VM and has no parent. One other important rule: a child can delegate to its parent but a parent can't delegate to any of its children. Consequently, a ClassLoader can look for a class in its parent but not in any of its children.

In adapting the JVM to the delegation model, three new ClassLoaders were introduced (see Figure 1). The first ClassLoader to be created is the bootstrap one. As previously stated, it's implemented in the VM and has no parent. Its role is to load core JDK classes found in the JAR files in the lib subdirectory of the JDK distribution tree (or set with the -Xbootclasspath parameter). The extension ClassLoader, which is written entirely in Java, loads from the directories specified by the system property java.ext.dirs. The default value for this property is the lib/ext subdirectory found in the JDK distribution. Its role is to load classes that extend the capability of the core JDK distribution. As shown in Figure 1, the bootstrap ClassLoader is the parent of the extension ClassLoader. Finally, there's the application ClassLoader whose parent is the extension ClassLoader. Its role is to read classes from the system classpath (the property name is java.class.path).

The delegation model causes core JDK classes to be loaded in isolation from the extension classes, which are in turn loaded in isolation from application classes. This is a great discovery, as it would appear that I could use ClassLoaders to isolate an application's classes. In addition, constructing a ClassLoader on demand will allow me to extend the classpath on demand. Now the question is: Do I buy or do I build a ClassLoader? I much prefer to buy, so let's see what the JDK has to offer in the way of ClassLoaders.

public class URLClassLoader
extends SecureClassLoader
This class loader is used to load classes and resources from a search path of URLs referring to both JAR files and directories. Any URL that ends with a "/" is assumed to refer to a directory. Otherwise, the URL is assumed to refer to a JAR file, which will be opened as needed.
The AccessControlContext of the thread that created the instance of URLClassLoader will be used when subsequently loading classes and resources.
The classes that are loaded are, by default, granted permission to access only the URLs specified when the URLClassLoader was created.
Since JDK 1.2

FIGURE 2 JDK URLClassLoader documentation

Java 2 ClassLoaders (Part 2)

At the top of the ClassLoader hierarchy sits the abstract class java.lang.ClassLoader. It contains all the behavior needed to convert an array of bytes into an instance of a class. It has one known subclass, java.security.SecureClassLoader. The SecureClassLoader adds support for the interactions with CodeSource, ProtectedDomain, and SecurityManager. A few of the services they provide include ensuring that class files are loaded by trusted sources only and that the bytecode has not been tampered with. SecureClassLoader has one known subclass, URLClassLoader. Figure 2 provides the Javadoc for this class.

This is great; the JDK comes with a ClassLoader that suits my needs. I can use the URLClassLoader to dynamically extend the system's classpath and, in the same stroke, isolate each application from any others that may be running. To unload an application I can simply unload the ClassLoader. In doing so, I cause no harm to other applications that may be running. Now that I have found a suitable ClassLoader, I can take the first step toward creating an abstraction of a Java application.

The Application Class

Let's start this exercise by writing down a description of a Java application. The classical Java application is a collection of one or more Java classes that cooperate to carry out some important function. As was previously alluded to, the process of launching a Java application can be broken down into four steps:

1. Finding, then loading, the class that implements the standard entry point, main
2. Finding, then invoking, the main method in the previously mentioned class
3. Waiting for the application to complete
4. Catching and reporting on any exceptions that the application lets slip through

This list specifies the behaviors that need to be implemented. Let's implement them and the supporting state one step at a time into a class called Application.

The first step states that the target class needs to be located and loaded. The code snippet demonstrates the technique commonly used to dynamically load a class.

```
Class targetClass = Class.forName(entryPointClassName);
```

A quick look at the Javadocs reveals that the method forName() loads the target class using the application

Rational Software

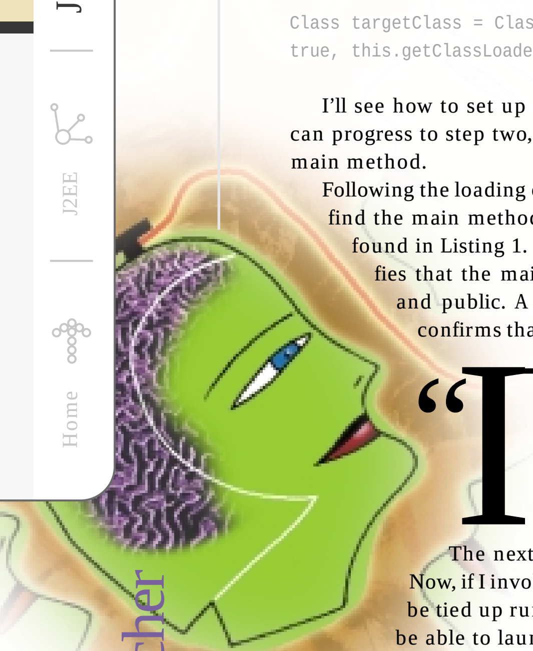
www.rational.com/offer/javacd2

ClassLoader. But I already know that none of our application classes will be on the application classpath. This call will throw a ClassNotFoundException. Okay, I've written only one line of code and already there's a problem, great! Let's press onward. Digging a little further into the Javadoc reveals that forName has been overloaded in the JDK 1.2. The new signature is forName(String className, boolean runStaticInitializer, ClassLoader classloader). This method finds the class named by className using ClassLoader. After the class is loaded, the Boolean determines if the static initializer should be executed. Since each application will have its own ClassLoader, using the new method signature allows us to specify it instead of using the system ClassLoader. Add a field ClassLoader and the supporting accessors to the application. The new code fragment solves the classloading problem.

```
Class targetClass = Class.forName( entryPointClassName,
true, this.getClassLoader());
```

I'll see how to set up the ClassLoader later on. For now, I can progress to step two, the discovery and invocation of the main method.

Following the loading of the main class, reflection is used to find the main method. The code for resolveMain() can be found in Listing 1. The JDK tools documentation specifies that the main method should be declared static and public. A check of the method modifier flags confirms that the correct method has been found.



"It's not often that I get to use the two terms *added flexibility* and *added security* in a positive way in the same sentence but that's exactly what the Java 2 class loading model offers"

The next step is to invoke the main method. Now, if I invoke main right away, my thread would be tied up running the application and I wouldn't be able to launch another application. I can avoid all this by executing each application in its own thread. Semantically, an application is a runnable entity, so it makes sense to have the application implement the Runnable interface. The Runnable interface requires that I implement a run method. If the main method is invoked in the run method, the main thread will be free to launch other applications.

Next question: What happens if the application throws an exception? The run method doesn't include any support for this, but it's actually a good thing because any application that throws an exception would force the launcher to handle it immediately. A more flexible approach is to have the application catch and store any exception that has been thrown. The launcher can then deal with the exception on its own terms. You can find the run method in Listing 1.

The run method includes behavior to help the application keep track of the running state. There are several ways I can do this. I can ask the thread if it's still alive. I don't care if the thread is running, I care if the application is running. Though one implies the other, it still seems more sensible to ask the application if it's running. The monitor would have to ask the application for the thread and I don't feel comfortable exporting internal state unless I have to. In this case, there seems to be no need to export the thread from the application. Thus, I've added a Boolean running to the application. Though access to and the setting of Booleans is guaranteed to be atomic, I still use synchronized to normalize all access to running. This becomes more important later on when setRunning starts to play an important role in thread control. When the run method ends, I always want running to be set to false. Delegating that method call to a finally block ensures that this always happens.

Let's give the responsibility for launch to the application. Since all the launch data is contained in the application, it only makes sense to put the launch method there. Again, the launch method is shown in Listing 1.

The launch method accepts a string array as arguments to be passed to the main method. The problem is, the run method doesn't accept any arguments. Adding an args field to the application solves this problem. The method goes on to define a thread for the application. After setting the context ClassLoader (to be explained later), it calls start() and the application is finally running. Now I'm left with implementing the last two steps.

First, I need the main thread to wait for the application to complete before letting it exit. Second, I need to report on any exceptions that may have been thrown and not caught by the application. These steps will be delegated to the class that created the application and then called the launch method. The private method setRunning (see Listing 1) sets the value for the field. If the value is set to false, it makes a call to notifyAll(). This releases any thread that may be blocked in the waitFor() method.

The Context ClassLoader

In the launch method, I took the mysterious step of setting the thread's context ClassLoader. Now I'll discuss how this fits into the delegation model and subsequently MAL. Recall from our discussion of ClassLoader chaining that a parent couldn't see classes in a child ClassLoader. It turns out that this restriction would be a showstopper if it were not for the context ClassLoader.

Why did I load each application in its own ClassLoader? This isolates the application and although this isolation is desired, it can also be a hindrance. Consider the case in which a class in a parent ClassLoader references a class's child. For example, I may use a component such as an O/R mapping tool loaded on the system classpath to persist an application class. By design, the applications classes are not visible to the O/R mapping tool. Consequently, attempting to reference one will cause a ClassNotFoundException to be thrown. I also must be careful if I add the tool to the application's classpath, as the ClassLoader plays a role in object identity. The same classes loaded from the same source into two different ClassLoaders will be considered different classes. Performing an operation in the wrong ClassLoader may cause a (terribly confusing) ClassNotFoundException to be thrown (or worse).

Each of these scenarios suggests that I need something more: a ClassLoader that functions within the application's context. It's not often that I get to use the two terms *added flexibility* and *added security* in a positive way in the same sentence but that's exactly what the Java 2 class loading model offers. By adding the field contextClassLoader and appropriate accessors to the class thread, the designers help support the isolation (and hence security) I require without breaking dynamic class loading schemes.

By default every new thread takes on the contextClassLoader of the thread it was created in. In my case, the contextClassLoader of the thread that creates an application thread is the application ClassLoader. But the only ClassLoader that

Dice.com

www.dice.com

FIGURE 3 System ClassLoader hierarchy

knows how to load an application's classes is the application's ClassLoader. If any component needs to load an Application class, it will need access to the application ClassLoader. Considering that the most likely thread to trigger the loading of an Application class will be the one that was created by the application, this works out perfectly. If you inspect the method jdkJava2Style in TestComponent (see Listing 2), you'll notice that it passes the context ClassLoader from the current thread into the forName method. This allows forName to load the class using the application ClassLoader. In effect, you've given the parent a controlled peek into one of its children.

Completing the Isolation of the Application's Execution Environment

The constructor for a new URLClassLoader takes an array of java.net.URL as an argument. The classpath for the ClassLoader is derived from this URL array. Note that once instantiated, the classpath for a URLClassLoader can't be changed. Since the class URL is instrumental in this application, it warrants closer scrutiny.

The constructor for the URL class accepts a string representation of the URL. The salient point here is that URLClassLoader assumes that the URL points to a directory or a JAR file. All paths ending with a "/" are assumed to be a directories. All others are assumed to be JAR files.

The constructor for an application uses a URL array to define the classpath for its ClassLoader. By placing the responsibility to construct an array of well-formed URLs on the caller, I haven't imposed any restrictions on how the information is to be collected.

Finally, any application that calls System.exit() or Runtime.halt() will cause the VM to exit. Clearly, I need some way to prevent this from happening. Fortunately, before the main body of these methods is executed, it checks with the SecurityManager to ensure that the caller has permission to make that call. Next I'll investigate how how to employ the SecurityManager to prevent applications from shutting down the VM.

The Java 2 Security Model

The Java 2 security model is a large, complex topic that falls well outside the scope of this article. I'll introduce only a few relevant portions of the subject.

As of the JDK 1.2, code must be granted permission to access a system resource. To ensure permission has been granted, JDK classes call the check methods found in the class java.lang.SecurityManager. Looking deeper into the documentation I found the runtime permission, exitVM, that controls VM shutdown. Wow, can I solve this problem by configuring this runtime permission? Well, if I make this change, neither the application nor MAL would be able to call System.exit(). I need to do something else. Again on the buy/build trail, I looked through the JDK to see what I might learn or reuse. This time, my search failed to turn up a suitable candidate, so that leaves the build option.

Constructing a SecurityManager

In the documentation for SecurityManager is a description for a checkExit method. It checks to see if the code has been granted the exitVM permission. If I override this method, I can check to see whether or not its MAL or an application made the exit call. The implementation of my specialization of SecurityManager can be seen in Listing 3.

MAL makes a call to System.setSecurityManager in a static initializer to install the specialized SecurityManager. There still are some interesting side effects that travel with this solution. For instance, once installed, a SecurityManager can't be replaced. That's good because I don't want an application to install its own SecurityManager. However, it's also bad because if any application does try to install a SecurityManager, it will fail. Could this be one type of application that MAL may not be able to deal with?

Running a Test Application

I crafted a small test application that consists of a component and an application. The component defines the interface TestComponentInterface and class TestComponent. The application contains a single class named TestApplication1, which implements TestComponentInterface. TestComponent uses reflection to create a new instance of TestApplication1 and then calls the method defined in TestComponentInterface.

TestComponent defines two methods, jdk11Style() and jdkJava2Style(). As expected, the former uses the single argument Class.forName() and the latter uses the three-argument Class.forName() method. Each of these methods accepts two parameters: one specifies the class implementing TestComponentInterface, and the other is used as an argument for TestComponentInterface.run(). A test application satisfying these requirements was crafted. The test calls both jdk11Style() and jdkJava2Style(). Running the tests requires that TestComponent be included in TestApplication's classpath. Both jdk11Style() and jdkJava2Style() run as expected. I haven't yet justified the need for the more complex three-argument Class.forName() method. This will be justified later on.

The first time I ran a test, I encountered a SecurityException (see Listing 4). Okay, I'm only overriding checkExit so what's the problem? I replaced MultiApplicationSecurityManager with SecurityManager and reran the tests. Same result. Okay, this helps because I now know that my security manager is not the source of the problem.

	Single VM (KB)	Second VM (KB)	Third VM (KB)	Total Memory Requirements (KB)
Single VM running one application	377.28	na	na	377.28
Two VMs, each running one application	375.6	230.5	na	606.1
Three VMs, each running one application	386.1	238.3	227.5	851.9
Single VM running three applications	384.1	na	na	384.1

TABLE 1 Test case memory requirements

ParaSoft Corporation

www.parasoft.com/jdj4

As it turns out, by default, a SecurityManager is not installed. When this happens, all check methods pass. If a SecurityManager is installed, its default behavior is to grant the permission specified in the files java.home/lib/security/java.policy and user.home/.java.policy. Well, there was no .java.policy file in my user.home directory and the former file grants only a limited number of permissions. To fix the problem, I used the policytool to create a .java.policy file (see Listing 5). Now my tests run as expected.

To see how much memory can be saved, let's first see how much memory is normally consumed using the traditional launcher. Four tests were defined. The first three tests execute one, two, and three instances of the test application, each in its own VM. The last test executes three applications in the same VM using the MAL. Each test was run three times in the JDK 1.3.0 on my 128MB W2K laptop. I configured the system tool perfmon to monitor the free memory counter.

Since I know that the OS shares resources, the tests needed to account for the effects of sharing. The first test established a baseline memory usage. Tests 2 and 3 provided the incremental amount of memory required to support the running of a second and third instance of the VM. Table 1 illustrates the memory requirement of each test.

Figure 3 provides a graph of the free memory counter over time. The graph displays the results of two different runs. I first ran three applications, each in its own VM. After free memory had returned to prerun levels, I then ran three applications all in the same VM. Each downward step is the result of a new VM grabbing a chunk of memory. It's interesting to note that in the first run, memory is consumed in three steps (one for each VM start) and recovered in four. In addition, the memory requirements of the first VM are greater than that of the two successive VMs. As each VM halts, memory is returned to the OS. After all the VMs have exited, the OS is now free to recover the shared text segment. Once this is completed, free memory returns to prerun levels.

In the latter run, we can see the expected decrease in demand for memory. Once again, witness the OS recovering the shared text segment as a separate step.

Though the technique does save memory, the results are not as big as I'd hoped. But there's another component to consider, time. By having a VM prestarted, it should be much quicker to launch applications. I'll leave the verification of that assertion as an exercise for the reader. I'll spend the remainder of this article spinning MAL through a second iteration.

Sharing Components

Recall that during the discussion of the contextClassLoader, the use of third-party components such as O/R mapping tools was brought up. It's common for these tools to use a pluggable integration strategy. In the current implementation of the launcher, I must choose between placing third-party components on the system classpath or on the application classpath. If I choose the system classpath, then applications can share it, but I won't be able to use multiple versions of the component nor will I be able to dynamically define it. If I wish to dynamically define the component, I need to place it on the application classpath. In making this choice, I've eliminated the possibility of sharing the component. Can I have my cake and eat it too? Can I dynamically define a sharable component? Stay tuned.

First let's consider some of the common elements and differences between a component and an application. An application is runnable; a component is not. An application must support the standard entry point. A component defines its own entry points, which may or may not be set according to a standard. An application is launched; a component is loaded. Applications and components can be loaded into their own ClassLoader and each requires its own classpath entry.

I'll use this initial set of observations to see if I can motivate a refactoring of MAL to include a Component class. Since the application looks like a specialization of the component, I'll start by having the application extend the component. As before, the application still implements Runnable and it maintains the behavior to monitor and control threading. All of the classpath and class-loading behavior can be pushed up into the component superclass. Once I finish the refactoring, I can focus on what new behavior needs to be added.

What's new in this version of the launcher is that applications and components can be dependent on other components. To gain a deeper understanding of what this means, consider the following example: I have an application, A, that uses a third-party O/R mapping tool, C. If I load both A and C in different ClassLoaders, I need to somehow ensure that A and C can see each other's classes. Chaining the ClassLoader containing A to the one containing C can solve the first half of the problem. If you look to the constructor in the component and the application, you can see an extra parameter has been added. The body of the constructor in the component contains the extra logic needed to create a ClassLoader with a parent other than the system ClassLoader.

To make A visible to C, I need to consider how C interacts with A. The interactions between TestComponent and TestApplication represent a typical interaction. In the jdkStyle methods, TestComponent tries to dynamically load TestApplication. In the jdk11Style method, first the component's, then the system's, ClassLoader is used. But I know that these ClassLoaders aren't visible to TestComponent so the method will throw a ClassNotFoundException (see Listing 6). In the jdkJava2Style method, the first ClassLoader contacted is the thread's context ClassLoader. In this case, the thread was started in Application.launch(). If we look back at that method, we can see that the thread's context ClassLoader is set to that of the applications'. Since the jdkJava2Style method uses the threads context ClassLoader, our problem is solved – C is visible to A. The jdkJava2Style method runs without incident.

The ComponentCache

The ComponentCache is a singleton that defines a repository for components. It's necessary to "pin" components into memory for the period of time when they're not referenced by anything else. The ComponentCache also acts as a "database" of components. This means that once loaded, the component will remain in memory for the lifetime of the VM. This brings up a number of interesting GC-related questions, which I won't talk about here.

"I could rewrite the System class to remove some of this contention, but the Java licensing agreement (for good reasons) would prevent me from distributing these modifications"

InstallShield Software Corp.

www.installshield.com

Further Refinements

A number of further refinements could be made to improve the launcher. The first would be to use a ThreadGroup to contain each application's threads into a single bundle. Second, the struggle with the SecurityManager clearly demonstrated that static and singleton implementations can wreak havoc with multithreaded applications. The System class contains a number of statics and singletons that reinforce assertion. Although I use singletons (as evidenced by ComponentCache), I tend to view them with a degree of skepticism. Is there something missing from the design? This is the first question that pops into my mind when I see one.

The System class allows access to a number of potentially nonshareable resources. Is there a design element that's missing? It's clear that MAL places different strains on the JDK than those originating from the traditional launcher. In a sense, MAL is trying to be an execution context. In this role, MAL falls short in several key areas.

The first shortcoming can be seen with the SecurityManager implementation. Though the Java 2 security model is much improved, it still doesn't totally eliminate the need for a customized SecurityManager. Applications that attempt to install their own SecurityManager won't be able to do so. I could have implemented a more complex custom SecurityManager, but this wouldn't have been necessary if I had an execution context in which I could install a regular SecurityManager. The next shortcoming is with System.properties.

It's quite common for applications and components to set properties in the System class. You could argue that if developers take care when naming properties, name-space collisions could be avoided. What happens when you wish to run two different versions of the same application? Different versions of the same application will most likely use the same property names. Again, I could provide an implementation that segregates properties by application, but a better solution would be to have these properties isolated into separate execution contexts.

Next, I'll talk about System.in, System.out, and System.err. I find exceptions difficult enough to deal with without having the added challenge of collating the output from several different applications. It would be much better if each application had its own console. Yet another problem: What if an application tries to install its own in, out, or err class? This problem is not so easily solved, as it implies MAL may have to make changes to a system.

I could rewrite the System class to remove some of this contention, but the Java licensing agreement (for good reasons) would prevent me from distributing these modifications. Most vendors that do require changes to base classes (some profilers modify Object) use a bytecode injection to distribute the change without violating the license agreement. This is a fairly complex technique that requires the use of a specialized ClassLoader. These are only a few of the traps of using statics and singletons in a multithreaded environment.

Summary

I've shown how you can construct a launcher to execute multiple applications. As demonstrated, this technique did reduce the overall demand for system resources. In introducing this technique, I've also introduced a practical application of the Java 2 ClassLoader model. This model is gaining wider acceptance as demonstrated by J2EE container implementers and, most recently, by IBM's new IDE project, Eclipse. Eclipse will load each plug-in into a separate ClassLoader. I've also learned to be more careful when posting advice to a discussion group.

kirk@javaperformancetuning.com

Author Bio
Kirk Pepperdine has more than 10 years of experience in OO technologies. In addition to his work in the area of performance tuning, Kirk has focused on building middleware for distributed applications.

Listing 1

```
package com.jpt.mal.application;

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

public class Application implements Runnable {

  protected URLClassLoader classLoader;
  protected URL[] classPath;
  protected String className;
  protected Object[] args;
  protected Exception applicationException;
  protected boolean running;


  public Application(String aClass, URL[] aClassPath) {
    className = aClass;
    classPath = classPath;
    running = false;
    classLoader = new URLClassLoader(
        classPath,
        Thread.currentThread()
.getContextClassLoader());
    args = new Object[1];
  }

  public Exception getApplicationException() {
    return applicationException;
  }

  public void launch(String[] args) {
    if (isRunning()) return;
    this.args[0] = args;
    applicationException = null;
    Thread thread = new Thread( this,
                       "Running " +
                       className);
    thread.setContextClassLoader(this.classLoader);
    thread.start();
    Thread.yield();
  }

  // Runnable targets
 public void run() {

    setRunning(true);
  try {
      this.resolveMain().invoke(null, args);
    } catch (Exception ex) {
      applicationException = ex;
    } finally {
      setRunning(false);
    }
  }

  private Method resolveMain()
    throws ClassNotFoundException,
    NoSuchMethodException {
    Class targetClass =
     this.classLoader.loadClass(className);
    Class[] arg_types = { String[].class };
    Method main = targetClass.getMethod("main", a
     rg_types);
    int mask = Modifier.STATIC ^ Modifier.PUBLIC;
    if ((main.getModifiers() & mask) != mask)
      throw new NoSuchMethodException(
      "Cannot find method public static void
main(String[]) in class "+ className);
      return main;
    }

  private synchronized void setRunning(boolean value) {
    this.running = value;
    notifyAll();
  }

  public synchronized boolean isRunning() { return run
   ning; }
```

ILOG

www.ilog.com/jdj/jrules

```
  public synchronized void waitFor() {
    while (isRunning())
      try {
        wait();
      } catch (InterruptedException ie) {}
  }
}
```

Listing 2

```
package com.jpt.testcomp;

public class TestComponent {

  public TestComponent() {}

  public void runJava2Style(String implementation, String arg)
    throws ClassNotFoundException,
    InstantiationException,
    IllegalAccessException {

    Class clazz = Class.forName( implementation, true,
                Thread.currentThread().getContextClassLoader());
    TestComponentInterface tci = (TestComponentInterface)
                             clazz.newInstance();
    tci.run(arg);
  }

  public void runJDK11Style(String implementation, String arg)
    throws ClassNotFoundException,
          InstantiationException,
          IllegalAccessException {
    Class clazz = Class.forName( implementation);
    TestComponentInterface tci = (TestComponentInterface)
                             clazz.newInstance();
    tci.run(arg);
  }
}
```

Listing 3

```
public class MultiApplicationSecurityManager extends
SecurityManager
  throws SecurityException {

public void checkExit(int status) {

  if ( Thread.currentThread().getContextClassLoader() !=
     ClassLoader.getSystemClassLoader())
    throw new SecurityException();

  super.checkExit(status);

}
```

Listing 4

```
java.security.AccessControlException: access denied (java.lang.Runtime
   Permission createClassLoader)
 at java.security.AccessControlContext.checkPermission(AccessControlContext.
   java:272)
 at java.security.AccessController.checkPermission(AccessController.
   java:399)
 at java.lang.SecurityManager.checkPermission(SecurityManager.java:545)
 at java.lang.SecurityManager.checkCreateClassLoader(SecurityManager.
   java:610)
 at java.lang.ClassLoader.<init>(ClassLoader.java:203)
 at java.security.SecureClassLoader.<init>(SecureClassLoader.java:56)
 at java.net.URLClassLoader.<init>(URLClassLoader.java:82)
 at com.jpt.application.Application.<init>(./src/com/jpt/application
   /Application.java:20)
 at com.jpt.ui.JShell.launch(./src/com/jpt/ui/JShell.java:60)
 at com.jpt.ui.JShell.main(./src/com/jpt/ui/JShell.java:72)
Exception in thread "main" Process terminated with exit code 1
```

Listing 5

```
/* AUTOMATICALLY GENERATED ON Sun Feb 03 15:43:44 EST 2002*/
/* DO NOT EDIT */

grant {
 permission java.security.AllPermission;
};
```

Listing 6

```
Running JDK 1.1 Style on com.jpt.testapp.TestApplication1
Expected
Exception:com.jpt.testapp.TestApplication1java.lang.ClassNotFoundEx
ception: com.jpt.testapp.TestApplication1
 at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
 at java.security.AccessController.doPrivileged(Native Method)
 at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:297)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:253)Running
JDK 1.1 Style on com.jpt.testapp.TestApplication2
 at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:313)
 at java.lang.Class.forName0(Native Method)
 at java.lang.
Expected Exception:com.jpt.testapp.TestApplication2
Class.forName(Class.java:120)
 at com.jpt.testcomp.TestComponent.runJDK11Style(Unknown Source)
 at com.jpt.testapp.TestApplication1.start(Unknown Source)Running
Java2 Style oncom.jpt.testapp.TestApplication1
 at com.jpt.testapp.TestApplication1.main(Unknown Source)
 at java.lang.reflect.Method.invoke(Native Method)
 at com.jpt.mal
com.jpt.testapp.TestApplication1:0 sleeping
.component.Application.run(Unknown Source)
 at java.lang.Thread.run(Thread.java:484)
java.lang.ClassNotFoundException:
com.jpt.testapp.TestApplication2Running JDK 1.1 Style on
com.jpt.testapp.TestApplication3
 at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
 at java.security.AccessController.doPrivileged(Native Method)
 a
Expected Exception:com.jpt.testapp.TestApplication3
t java.net.URLClassLoader.findClass(URLClassLoader.java:188)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:297)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:253)
 at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:313)
 at java.lang.Class.forName0(Native Method)
 at java.lang.Class.forName(Class.java:120)
 at com.jpt.testcomp.TestComponent.runJDK11Style(Unknown Source)
 at com.jpt.testapp.TestApplicaRunning Java2 Style
oncom.jpt.testapp.TestApplication2
com.jpt.testapp.TestApplication2:0 sleeping
tion2.start(Unknown Source)
 at com.jpt.testapp.TestApplication2.main(Unknown Source)
 at java.lang.reflect.Method.invoke(Native Method)
 at com.jpt.mal.component.Application.run(Unknown Source)
 at java.lang.Thread.run(Thread.java:484)
java.lang.ClassNotFoundException: com.jpt.testapp.TestApplication3
 at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
 at java.security.AccessController.doPrivileged(Native Method)
 at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:297)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:253)
 at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:313)
 at java.lang.Class.forName0(Native Method)
 at java.lang.Class.forName(Class.java:120)
 at com.jpt.testcomp.TestComponent.runJDK11Style(Unknown Source)
 at com.jpt.testapp.TestApplicaRunning Java2 Style
oncom.jpt.testapp.TestApplication3
com.jpt.testapp.TestApplication3:0 sleeping
tion3.start(Unknown Source)
 at com.jpt.testapp.TestApplication3.main(Unknown Source)
 at java.lang.reflect.Method.invoke(Native Method)
 at com.jpt.mal.component.Application.run(Unknown Source)
 at java.lang.Thread.run(Thread.java:484)
com.jpt.testapp.TestApplication1:1 sleeping
com.jpt.testapp.TestApplication2:1 sleeping
com.jpt.testapp.TestApplication3:1 sleeping
com.jpt.testapp.TestApplication1:2 sleeping
com.jpt.testapp.TestApplication2:2 sleeping
com.jpt.testapp.TestApplication3:2 sleeping
com.jpt.testapp.TestApplication1 done!
com.jpt.testapp.TestApplication2 done!
com.jpt.testapp.TestApplication3 done!
```

Download the Code!
www.JavaDevelopersJournal.com

Actuate Corporation

www.actuate.com/info/jbjad.asp

Prescriptions for Your Java Ailments

Ask Doctor Java

WRITTEN BY
JAMES MCGOVERN

Send your questions, praise, comments, and admiration to doctorjava@sys-con.com.

Published letters will be edited for length and clarity. Any reference to third parties or third-party products should not be construed as an endorsement by Doctor Java or *Java Developer's Journal*.

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

Q **I'm constructing a tree from a file and I need a way to halt the tree construction until after the file is finished reading from a separate thread.**

A A couple of approaches come to mind. Your problem centers on the behavior of threads, so let's look at a couple of possibilities. First, you could consider having the TreeBuilder thread check a global variable such as isFileLoadComplete and yield if file loading has not been completed. You could have the FileLoad thread get a reference to the TreeBuilder thread and make it sleep until the file has been loaded. You could also use thread synchronization using notify and waitFor. I recommend none of those approaches, and instead suggest a different design.

If you want to build a tree, it's easier to do it in the same thread. Consider the following approach: upon opening a file, create an instance of the tree. When reading each line of the file, update the tree in place. Once you're done reading the file, copy the tree to its appropriate location. This approach will keep your code clean and allow your program to run faster. If the tree you're adding to is also updated by simultaneous file reads, make sure that you wrap it in a synchronized block.

I hope this solves your problem.

Q **In your January column, (*JDJ*, Vol. 7, issue 1), your answer to the question about authentication to a Windows 2000 server gave me some good insight. Our company has chosen not to go to Active Directory yet. Can you describe a similar approach to the problem using Windows NT 4.0 server?**

A Active Directory is one of the better implementations of LDAP on the market and is one of the few that isn't licensed on a per-entry basis. I hope your organization will migrate in a timely manner. To authenticate to a Windows NT domain, you can't use the JNDI approach mentioned but will need to consider implementing the Java Authentication and Authorization Service (JAAS). JAAS provides a Windows NT provider that will meet your needs. For more information on JAAS visit http://java.sun.com/products/jaas/index-10.html.

Q **I've been doing a little reading on JAAS and I like the fact that there's a standard API for authentication and authorization. Can you describe how to use JAAS's authentication mechanism without relying on its authorization machinery? For example, I'd like to use JAAS as the standard way to find out who the current client is and then use that knowledge to make some minor behavioral adjustments to the code. For example, if you're logged in as Bob, then I might tweak a SQL statement a bit so you don't see as much as user Ted. Can JAAS do this for me?**

A Let me make a couple of assumptions: first, if you have your own authentication mechanism, such as a login page, you most likely will have stored who the user is in a bean. If you want to know who the logged-on user is in a Swing application, you can determine this from the Java system properties that have a user.name attribute.

JAAS wouldn't help you with tweaking SQL statements, as this is better handled by either database-level security such as creating a new JDBC connection for each user or by delegating the change of SQL to a stored procedure. JAAS would be appropriate for authentication and authorization if you were integrating with a third-party security provider, such as the Active Directory and Windows NT Domain, or using a single sign-on solution such as Netegrity's SiteMinder, RADIUS Servers, or Kerberos.

Each of these products will require that the user be authenticated before any authorization step takes place. You can hide the authentication step within your application, but you can't skip it.

Q **I have a servlet that talks to a secure gateway to authorize credit cards. It uses JDK 1.3.1 and JSSE. When I upgraded to JDK 1.4, it stopped working. The following line of code throws a ClassCastException:**

```
HttpsURLConnection connection =
    (HttpsURLConnection).url.openConnection();
```

Could you tell me why my code doesn't work?

A This problem can be easily corrected by changing some import statements. In JDK 1.4 HttpsURLConnection has changed to javax.net.ssl and no longer uses com.sun.net.ssl. If you still need to use an older class, you can reference it at startup by using:

```
java–Djava.protocol.handler.pkgs=
    com.sun.net.internal.www.protocol
```

Alternatively, you can set it at runtime as follows:

```
System.setProperty("java.protocol.handler.pkgs"
    ,"com.sun.net.ssl.internal.www.protocol");
```

Q **What's the best way to internationalize exceptions?**

A I'm not sure of the best way, but I can show you how I would approach the problem. I would suggest creating your own exceptions and throwing them using internationalized strings. The fol-

New Atlanta Communications

www.newatlanta.com

lowing code snippet demonstrates the approach:

```
import java.util.Locale;
import java.util.ResourceBundle;

public class InternationalizedException extends
    Exception {
      ResourceBundle rb = Resource
            Bundle.getBundle("i18nBundle",
               Locale.getDefault());

   public InternationalizedException() {
      super(rb.getString("exceptionKey");
   }
}
```

To learn more about Java and internationalization, I suggest visiting the following URL: http://java.sun.com/doc/books/tutorial/i18n.

Q: I'm using Java and EJBs to access a legacy system. Accessing the host each time is very expensive. The data from the legacy system is read-only and I'd like to cache it somewhere. I'm not sure where it should be stored.

A: The Java Community Process is working on the JCache specification that will solve your problems in the future. In the meantime, let me see if I can help you with your problem. For EJB developers the first thought may be to store it in a stateful session bean. Stateful session beans would cause the construction of the bean for each client access, so this won't work. The next idea may be to use stateless session beans. This won't work either, because stateless session beans exist in a pool and the container determines when and how many instances will be constructed.

Entity beans may also come to mind but may not work well either. Remember that the EJB container can call activate and passivate anytime it desires. The activate call would need to retrieve the value from your legacy system again.

Some developers might consider using the Singleton pattern in this situation. This works in very limited situations and never works in a high-availability situation. As there is never a Singleton in a cluster, this pattern shouldn't be used. Furthermore, in Java a class is unique based on its ClassLoader, not the Java VM itself.

Let's discuss a couple of approaches that will actually work. First, you can consider using JNDI to store your data. In many of the application servers, JNDI is a clusterable resource, so changes to your read-only data will be replicated to all instances. A second approach is to use an RMI server where you can store copies of your data and have all EJBs connect to it. For high-availability situations you'll need to make the RMI server aware of other RMI servers that provide the same function. A third approach is to store your cached data in an object-oriented database such as Gemstone or Versant. Many databases optimize I/O and cache the data in memory, so there won't be any round-trips between the database and application server.

Author Bio

Doctor Java, aka James McGovern, moonlights as an enterprise architect with Hartford Financial Services (www.thehartford.com), where he focuses on the architecture of high-availability and J2EE-based solutions.

Q: How do I set the referrer for a URLConnection?

A: The following code snippet demonstrates how to set the referrer:

```
URL url = new URL("http://doctor
   java.htsco.com");
URLConnection conn =
   url.openConnection();
conn.setRequestProperty("Referer","http://doc
   torjava.sys-con.com");

InputStream is = conn.getInputStream();
BufferedInputStream bis = new
   BufferedInputStream(is);
```

Of course, you'll need to catch any MalformedURLExceptions that might be thrown.

Q: What is your opinion of Jtrix?

A: Jtrix is remarkably similar to Jini, as both offer a distributed-service model. Jtrix added some useful features that don't currently exist in Jini, such as a hosting service where a service can find somewhere else to run and copy itself into that space. This will allow a Jtrix application to scale almost endlessly, as it will distribute itself as needed. I can see this being a powerful feature for Web services that in the future may need to be accessed across the planet.

Jtrix added accounting features as well. If a Web service needs to expand via its hosting service, the provider will need a mechanism to bill based on resource usage (CPU, memory, network traffic, and so on). Lack of an accounting feature will hinder the growth of all service-based approaches. The other feature that Jtrix provides is the complete separation of client and server. In Jini the client and server have to use the same versions because they're in the same space.

I prefer that developers use Java standards where they exist and would hope that the Jtrix team would formalize their work using the Java Community Process and focus on making Jini better. If you decide to use Jtrix, I suggest that the Jtrix netlet export a Jini service as a default.

doctorjava@sys-con.com

SYS-CON MEDIA

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

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

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

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

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

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

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

CUSTOMER RELATIONS/JDJ STORE
MANAGER, CUSTOMER RELATIONS/JDJ STORE
ANTHONY D. SPITZER tony@sys-con.com

LOOX
Software Inc.

www.loox.com

JASON R. BRIGGS J2ME EDITOR

A Long Way to Go

A strange accident occurred on my flight back to New Zealand. Somehow, the plane flew through a rip in space-time and we wound up in a freak alternate dimension. The thing is, it was initially very difficult to tell that we weren't in the right dimension anymore, because everything was pretty much the same. But then I picked up an Australian computer magazine and read an editorial in which the writer, I'll call him Mr. KnowNothing, was talking about how Bill Gates had underestimated the Internet initially but eventually had turned Microsoft around to become a technology leader with its .NET strategies.

That's when I knew something must have happened on that 747 bound from Sydney to Auckland (and it explains the severe turbulence). Because in my universe, .NET was still in beta, and as far as I'm concerned, the battle certainly wasn't over.... Hell, we were just starting to warm up. As I think Alan has mentioned before (in this dimension he's world famous as the character Blane Evans, on "Days of our Lives"), all Microsoft has done is repackage ideas that have been around for a long time (a process I believe they refer to as "innovation"), which hardly qualifies them as a technology leader. But it seems, according to Mr. KnowNothing, in this dimension Microsoft has won the war.

In desperation I've invented a machine in my garage – mainly constructed from old computers, car stereo parts, the laser component from a DVD player, and a hyperactive hamster called Gerald (to provide power) – that will transport me back to my original dimension. My one worry is that if I miscalculate, I could end up in the dimension where Visual Basic became the only language for serious computer work, or (shudder) Modula-2. It doesn't bear thinking about really.

• • •

Since coming home I've had a vivid illustration of just how far Java has to go. Everyone knows Microsoft, of course. An almost microscopic percentage of those who don't work in the industry (i.e., one person) have even heard about .NET, but Java still brings blank faces, or thoughts of Indonesian islands and coffee to the majority (this is not to say that people in my small hometown are ignorant compared to Londoners, rather that when returning home, you are invariably asked by all and sundry what you've been doing for the past few years).

In the past, I haven't worried that Java wasn't too well known. In fact I've personally been hoping the profile stays low – at least in the wireless space – but that penetration of the market will become more widespread. Now I'm not so sure. As Rick Ross, president of JavaLobby, says, there are 200,000,000 reasons to take .NET seriously. And while you might not think that .NET will have any impact on J2ME, with a company that has the resources to spend that much money just on marketing, you do have to wonder.

Now I want Java logos on everything.

Just in case.

• • •

High-performance – not something you might typically associate with a MIDP application. But a press release from esmertec recently caught my eye with a project that might have a chance to change that, at least for Intel-based (PXA250 and PXA210) mobile devices. esmertec's announcement that they would be working with Intel Corp. to "deliver optimized Java solutions for high-performance mobile computing using the Intel XScale microarchitecture and Intel Flash Data Integrator (FDI)" is another of those flowery phrases that marketing people love littering press releases with. But it means that another big player (Intel), which hasn't generally been that visible in the Java world, is, at the very least, thinking about J2ME as a viable platform. Hopefully at the end of it, we'll see more devices on the market capable of running MIDP apps at better speeds. I'm still waiting for "MIDP Quake" on a mobile phone....no let me qualify that... "MIDP Quake" running at a high speed on a mobile phone. Go esmertec!

• • •

In this month's J2ME section, Dan Pilone talks about distributed computing in the J2ME world, while James White discusses some of the issues you should be thinking about before starting on a mobile or wireless application.

jasonbriggs@sys-con.com

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

QUALCOMM
Incorporated

www.brew2002.com

Pervasive Computing: the Next Generation of Consumer Applications

Leverage the new capabilities

WRITTEN BY
DAN PILONE

How enterprise software is written has undergone a major shift with the introduction of distributed technologies like EJBs and Web-based thin clients. However, this new approach to writing software has not trickled down to consumer applications such as recipe managers, cookbooks, or word processors.

I discuss some of the problems that historically have held back distributed development, present an example next-gen application using J2ME, and then discuss some of the architectural issues inherent in distributed consumer software.

Let's face it, there are few real consumer-level distributed applications out there. Traditionally, they're difficult to build. The tools and infrastructure needed to support and deploy *wired* distributed apps, let alone *wireless* ones, are hard to come by. CORBA 1.0 was introduced in 1991 and has found a home in enterprise development. Netscape made a valiant attempt to incorporate Visigenic's ORB into their product line, but failed to make CORBA commonplace on the desktop. Microsoft has gone through four component technologies, admittedly not all of them distributed, and is just beginning to make distributed computing a reality on the average user's desktop. Probably the most successful consumer-level distributed applications are the RC5 or SETI clients that are being used around the globe to look for E.T.

However, times have changed. Software developers now have tools that make designing and building distributed applications easier than ever. Wireless technologies and broadband in the home have helped meet the communication requirements. While technologies such as 3G wireless are still a little way out, developers should be planning for the changes that pervasive communications will introduce. There are still some psychological and technical hurdles to overcome, but hopefully the example application will put some of these to rest. Now the question is: "How can a developer leverage the available tools to make an application that's useful for the consumer?"

Users (developers and nondevelopers alike) tend to be task-focused. An application is a tool that gets the job done. The idea behind distributed consumer applications is to provide the user with this tool in a variety of different environments. The key is using the right tool for the right job. For example, a professional carpenter may use a nail gun to frame a house, but a regular hammer to build a tree house. The goal is to nail two pieces of wood together, but the tool varies depending on the environment. Thinking this way about consumer software opens up a whole new type of application development.

A Hands-on Demo

To demonstrate how easy it is to build a distributed application, this article presents a distributed text editor. It's a very simple example of a next-generation consumer application. The editor allows users to create, load, edit, and save text files. More advanced features, such as spell checking and laying out documents, are discussed from an architectural perspective, but not implemented. In the spirit of pervasive computing, users should be able to perform these functions on their data from anywhere using a variety of devices. The sample application will have a Swing-based desktop client and a J2ME-based medium client. All the tools used to develop, test, and deploy this application are freely available and referenced at the end of the article.

Before we get into the technical details of the implementation, let's nail down what we mean by distributed consumer applications:

1. Clear separation between the client and the "back end"
2. Transparent support for multiple types of clients
3. No requirement for a dedicated server
4. Different clients transparently accessing the same data

By following these principles, we can separate the tool (the client piece) from the task (various "back-end" functions). This follows the typical thin-client enterprise application pattern so far, but items 3 and 4 change things slightly. Our back end should not require a dedicated server, meaning we want to get away from the typical enterprise application server that supports large numbers of users. Instead, we want a lightweight, personal application server that supports a much smaller number of users. Traditional application server capabilities, such as failover or high-end scalability, are not needed for these simpler applications. Finally, the back-end func-

PointBase, Inc.

www.jdj5.pointbase.com

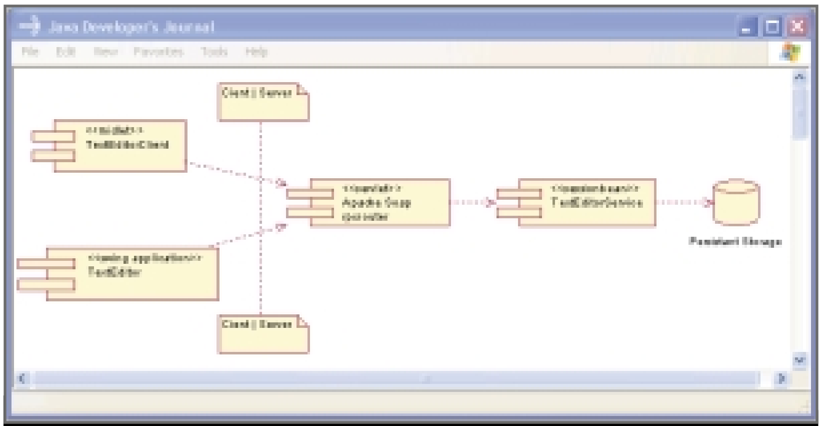


FIGURE 1 Text editor component diagram

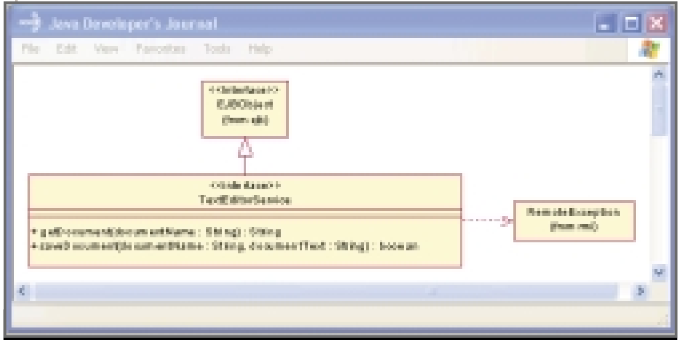

Java Developer's Journal
File Edit View Favorites Tools Help
<<Interface>>
EJBObject
(from ejb)
<<Interface>>
TextEditorService
+ getDocument(documentName : String) : String
+ saveDocument(documentName : String, documentText : String) : boolean
RemoteException
(from rmi)

FIGURE 2 TextEditorService remote interface

tionality operates on the same set of data regardless of the client being used. A user running the PDA client has access to the same data that was edited with the desktop client.

In the example application, this back-end functionality is provided by EJBs. EJBs make writing the distributed back end easy, however, they're not the ideal solution. Application servers require configuration and are much more comfortable on higher-end machines. A lightweight application server would be a better solution. Failover, clustering, and possibly even transaction support may be removed to help make a lightweight application server more desktop-friendly.

Client technologies are in better shape. Even if you restrict your selection of client platforms to Java, there are several options for scaling from the J2SE desktop client to the J2ME thin client. The sample application uses both to demonstrate transparent data accessibility.

Under the Hood

The application is a typical *n*-tiered application, in this case, two tiers. (Some would argue three, with persistent storage being the third. However, since we're simply saving information to the local file system with no DAO type pattern, I'm hesitant to call that a tier.)

The application uses the file system directly, rather than a database, in keeping with the consumer-level concept. Figure 1 shows the clean separation between the clients and the service-based back end

In a typical enterprise development process we would identify key use cases or user stories. Since we're trying to separate the "tasks" of the application from the GUI, we'll use this approach. To keep things simple, we'll stick with these four key tasks:

1. Create a new document
2. Load a document
3. Edit a document
4. Save a document

We'll use a stateless session bean, the TextEditorService, to provide the interface to our back end (see Figure 2).

Basic editing of the text document will be the responsibility of the client. Since we'll be accessing the file system directly from the TextEditor, implementing the load and save methods of the bean is easy. The getDocument method is shown in Listing 1.

As mentioned before, we'll provide two different clients, Swing-based and J2ME (see Figures 3 and 4, respectively).

The next architectural issue to be addressed is the communication between the tiers. Even if we were building a Java-only solution, directly accessing the TextEditorService bean would not be a viable option because of our J2ME client. Regardless, since we are designing a services-based application, we should try to stick to standards used in similar applications. We'll use SOAP to communicate with our services layer. SOAP clients are available for nearly every language on just about every platform, including J2ME. On the server side we'll use Apache SOAP to deploy our TextEditorService. Once Apache SOAP is up and running, we can deploy our service with the SOAP descriptor shown in Listing 2.

For simplicity, we'll be using the Apache SOAP RPCrouter as our servlet interceptor rather than providing our own as suggested by the Sun Blueprints.

For our desktop client we can simply use the client side of Apache SOAP. However, our J2ME client needs a special implementation due to tight memory and performance requirements. Fortunately, a great J2ME SOAP library is available from Enhydra.org, kSOAP. Built on top of kXML, the XML parser and SOAP client together total about 50K. The J2ME client was compiled and tested using Sun's J2ME Wireless Toolkit. Start ktoolbar, create a new project, and simply place the ksoap.jar and kxml-min.jar in the newly created lib directory. Any JARs placed in this directory are automatically added to the classpath when the toolkit compiles your application.

To hide the SOAP interface, we'll create a proxy class that mirrors our TextEditorService interface; however, the method implementations will use SOAP to make the actual calls (see Figure 5 and Listing 3). *Note:* In our proxy we hard-coded the URL to our SOAP interceptor. This obviously is not the ideal solution and should really be part of the deployment information. However, for the sake of simplicity, it's hard-coded in the source. Our J2ME application will simply make calls to the proxy, unaware that the call is actually being routed to our EJB service.

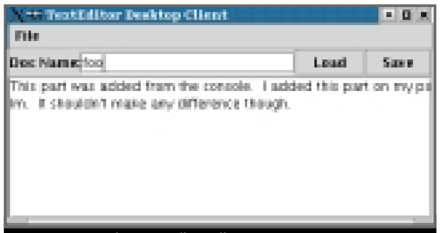
As seen in Listing 4, our J2ME client creates one form and one text editor widget. The form has space for the user to enter the file name and buttons to retrieve the document. When the document is successfully loaded, the editor is displayed until the user clicks Save or Cancel. Save executes a call to the proxy to store the document using the back end, and Cancel simply redisplays the original form.

The Swing client functions similarly: two JButtons to load and save the document, and a JTextArea to allow the user to edit the document. Again we're using a proxy class to wrap our SOAP interface. kSOAP can be used with J2SE if we use the SE transport, so we could reuse our existing J2ME proxy class (isn't Java wonderful?). However, we'll use the Apache client to demonstrate using a different SOAP implementation.

Once the clients are compiled, the Swing client can be started from the

Quintessence Systems Limited

www.in2j.com



foo

This part was added from the console. I added this part on my p
lm. It shouldn't make any difference though.

FIGURE 3 Swing TextEditor client

command line. Enter some text in the text area, enter a filename, then click Save. The Swing client then uses the proxy to make a SOAP call to the EJB back end. The TextEditorService will then save the file on the local file system. Now we can start the J2ME client, either through the Wireless Toolkit's emulators, or on an actual J2ME device such as a Palm VII or JavaPhone.

Type in the same file name and click Load. The J2ME client uses the EJB back end to retrieve the same document. While this is obviously a very simple application, it demonstrates the key concepts of distribution, multiple clients, and transparent access to back-end data. However, to some extent this is the easy part.

Architectural Issues

When discussing software architecture, key architectural drivers are often referred to as the "-ilities" – reliability, scalability, etc. Providing a distributed consumer application has some of the same "-ilities" as an enterprise application, plus a few that are somewhat unique to the consumer nature of the application. Below are guidelines to help address these issues and hopefully spur discussion when the solution is not clear.

Security

Security is obviously not unique to consumer applications, however, consumers are not expected to log in to each application before use. To make matters worse, distributed applications inherently make data available to external clients. Securing the communication channel is a must, and, thanks to the XML format used by SOAP, can be easily accomplished with a secure sockets layer (SSL), such as HTTPS to the back end. However, authenticating the user with the back end is more difficult. Ideally the user wouldn't need to be involved in the authentication using a public/private key type system. However, since these applications specifically target devices that can be misplaced or stolen, such as cell phones and PDAs, relying on device authentication is not an effective means of security. Since authentication will be needed for each application, a common framework should be developed. It's possible that projects, such as the Liberty Alliance project, may provide an authentication mechanism that could be used for consumer applications. However, this project is still in the formative stage and a solution is needed immediately.

Designing a security framework for distributed applications is beyond the scope of this article, however, a few key features may be identified:

1. Single sign-on for applications
2. Same authentication credentials regardless of client
3. A fully encapsulated security component that may be easily shared between applications
4. Support for communication channel encryption
5. No reliance on the underlying back-end component model

While these requirements are not unique, they need to be met on a wide range of platforms, including limited devices such as cell phones. A quick and dirty solution is to use HTTPS for SOAP communications and make use of the application server's credential support by using cookies to transfer session keys back to the client. J2ME clients can use the CLDC record support to persist this information between invocations. However, since records are not shared between J2ME applications, this is not an ideal solution. J2SE applications fare somewhat better in that a convention can be established to store credential information. As with a typical Web application, credential information will need to expire after a given period of time to avoid the problem of "authenticating the device, not the user."

Task Partitioning

Identifying the physical location of functionality is a key part of architecting a distributed application. The "thin-ness" of the client must be weighed against the ability to use the application without requiring a round-trip to the back end. It's conceptually possible to have thicker clients offering functionality not found on other clients. However, if this functionality is desired on one of the thinner clients, it must be replicated on the back end. A cleaner approach would be to provide as much functionality as possible in the back end and access it from the appropriate clients. Determining client "appropriateness" depends on several factors:

1. GUI and processing capabilities of the intended device
2. "Cost" of communicating with the back end, including time, availability, and possible financial costs
3. Amount of interaction required by the user
4. Intermediate store requirements placed on the client

By carefully partitioning functionality, it's possible to minimize the interaction and cost to the user. For example, spell checking may be added to our text editor. The spell-checking service may be written so the entire document is checked and the results, identifying which words were not found in the dictionary, are returned to the client. These words could be identified by their position in the document rather

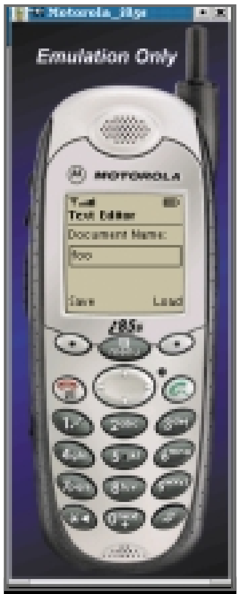
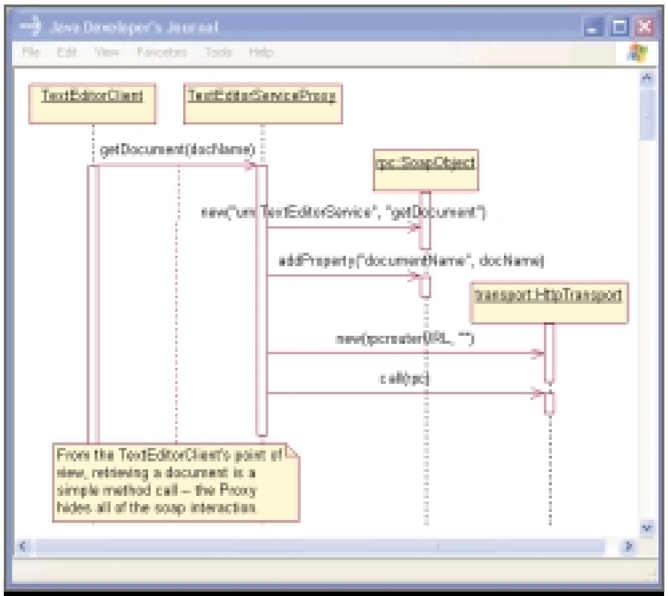


FIGURE 4 J2ME TextEditor client



From the TextEditorClient's point of view, retrieving a document is a simple method call – the Proxy hides all of the soap interaction.

FIGURE 5 getDocument Sequence diagram

Thought Inc.

www.thoughtinc.com

than the actual word to reduce the bandwidth required. However, this places the burden on the client to be able to find the identified word. On small memory devices this may not be possible. Once the word is identified, the user may correct it and the corrections are returned to the back end for integration.

Another example of properly partitioning tasks would be needed if we expanded our text editor to support several fonts and styles. Displaying and editing font information would be easy on a Swing-based client; however, it would be nearly impossible on a Java phone. Instead, the back end may provide services to retrieve the content separately from the markup information. Users can use the desktop client to produce their initial document, then make a few quick adjustments with their cell phone. The back end would retain markup information produced by the desktop client even though the cell-phone client can't make use of it.

Hardware Demands

While the example application used EJBs to provide back-end functionality, a lighter weight component model would be more desirable. More specifically, a lightweight application server could make desktop-based distributed applications more of a reality. As mentioned earlier, the scalability and fault tolerance requirements for a desktop application would be much less than that of an enterprise application. However, EJBs do make developing distributed applications relatively easy compared to building CORBA components. In addition to hardware requirements, automated installation and setup would be required. Installing a distributed application shouldn't require any more knowledge on the part of the user than a standard application. However, this places the burden on the application server and the application developer.

It would also be helpful for back-end developers to have a database at their disposal. Again, if properly configured and accessible from the application server, the user interaction would be kept to a minimum.

In all fairness, Microsoft has already provided most of this with their current operating systems, though not for the Java platform. It's time for the Linux distributors to step up to the plate and offer the user a similarly preconfigured set of back-end services.

Integration with Web Services

There has been much talk about Web services and .NET recently. Conceptually they're similar to the services-based, back-end approach outlined here. By leveraging SOAP between your clients and the back end, newly developed consumer applications are poised to integrate with Web services should they become widely available to the consumer.

In the meantime, applications may be developed and deployed without requiring a subscription model or integration with an external system. Desktop clients can be deployed immediately and will appear identical to applications currently available. As wireless devices become more prevalent, more and more application developers can release PDA thin clients to take advantage of previously installed back ends.

Conclusion

Now that broadband connections are becoming commonplace and wireless technology is becoming a reality, developers should begin positioning their applications to leverage the new capabilities. The tools to develop and deploy these new applications are freely available and quite mature. However, end-user support is much further behind.

Acknowledgment

I would like to thank three colleagues for their suggestions, discussions, and reviews: Michael Hudson, Christian Nelson, and Rich Newcomb.

References

1. *JBoss Application Server:* www.jboss.org
2. *Ant:* http://jakarta.apache.org
3. *Apache SOAP:* http://xml.apache.org
4. *kSOAP:* http://ksoap.enhydra.org
5. *J2ME Wireless Toolkit:* http://java.sun.com
6. *JDK 1.4b3:* http://java.sun.com
7. *Netbeans 3.3:* www.netbeans.org

pilone@slac.com

AUTHOR BIO

Dan Pilone is a software architect for Blueprint Technologies and an active contributor to the open source community.

Listing 1

```
/**
 * Returns the text of the document specified in documentName.
 */
public String getDocument(String documentName)
{
  System.out.println("getDocument(" + documentName +")");
  StringBuffer documentText = new StringBuffer();

  try
  {
      // We prepend /tmp to prevent creative people
      // from hosing our system...
      BufferedReader inStream =
        new BufferedReader(new FileReader("/tmp/JDJTemp/" +
      documentName));
      String tmp = inStream.readLine();
      while (tmp != null)
      {
          documentText.append(tmp);
          tmp = inStream.readLine();
      }
  }
  catch (IOException e)
  {
      documentText.append("<Error reading document>");
  }
  return documentText.toString();
}
```

Listing 2

```
<?xml version="1.0"?>
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
             id="urn:TextEditorService">
  <isd:provider
    type="org.apache.soap.providers.StatelessEJBProvider"
                scope="Application"
                methods="getDocument saveDocument">
    <isd:option key="JNDIName"
                value="TextEditorServiceHome"/>
    <isd:option key="FullHomeInterfaceName"
                value="com.pilone.texteditor.TextEditorServiceHome" />
    <isd:option key="ContextProviderURL"
                value="nova.slac.com:1099" />
    <isd:option key="FullContextFactoryName"
                value="org.jnp.interfaces.NamingContextFactory" />
  </isd:provider>
  <isd:faultListener>
    org.apache.soap.server.DOMFaultListener
  </isd:faultListener>
</isd:service>
```

Listing 3

```
import org.ksoap.*;
import org.ksoap.transport.*;

public class TextEditorServiceProxy
{
  public String getDocument(String documentName)
  {
    String document = null;
    try {
      SoapObject rpc = new SoapObject("urn:TextEditorService",
"getDocument");
      rpc.addProperty ("documentName", documentName);
      HttpTransport transport = new HttpTransport("http://www.sys-
con.com/soap/servlet/rpcrouter", "");
      document = transport.call(rpc).toString();
    }
    catch (Exception e) { e.printStackTrace(); document = e.toString();
}
    return document;
  }

  public void saveDocument(String documentName, String documentText)
```

Borland Software Corp.

www.borland.com/new/optimizeit/94000.html

```
  {
    try {
      SoapObject rpc = new SoapObject("urn:TextEditorService",
"saveDocument");
      rpc.addProperty("documentName", documentName);
      rpc.addProperty("documentText", documentText);
      HttpTransport transport = new HttpTransport("http://www.sys-
con.com/soap/servlet/rpcrouter", "");
      transport.call(rpc);
    }
    catch (Exception e) { e.printStackTrace(); }
  }
}
```

Listing 4

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;
import javax.microedition.io.*;

// SOAP imports...
import org.ksoap.*;
import org.ksoap.transport.*;


public class TextEditorClient extends MIDlet
  implements CommandListener
{
    public TextEditorClient ()
    {
 // Create the main form
 mainForm.append(m_DocumentName);
 mainForm.addCommand(m_GetCommand);
 mainForm.addCommand(m_SaveCommand);

 // Create the document viewing form
 m_DocumentText.addCommand(m_SaveCommand);
 m_DocumentText.addCommand(m_CancelCommand);

 // We'll handle button presses for both
 mainForm.setCommandListener (this);
 m_DocumentText.setCommandListener(this);
    }

    public void startApp ()
    {
 Display.getDisplay (this).setCurrent (mainForm);
    }

    public void pauseApp ()
    {
    }

    public void destroyApp (boolean unconditional)
    {
        // We could persist our current document
        // to a record store here if needed
    }

    public void commandAction (Command c, Displayable d) {
 // Find out which command was activated
 if (c == m_GetCommand)
 {
     // Retrieve the document name, then retrieve the
        // document from our proxy.
     String documentName = m_DocumentName.getString ();
     m_DocumentText.setString
   (m_TEProxy.getDocument(documentName));
     Display.getDisplay(this).setCurrent(m_DocumentText);
 }
 else if (c == m_SaveCommand)
 {
        // Ask our proxy to save the object,
        // then return to the main form
     m_TEProxy.saveDocument(m_DocumentName.getString(),
                            m_DocumentText.getString());
     Display.getDisplay(this).setCurrent(mainForm);
 }
 else if (c == m_CancelCommand)
 {
     // Simply redisplay the main form.  We can ignore
        // changes that were made to the document since
        // the only way to return to this screen is to
        // try to load a document that will refresh the
        // contents of our document variable
     Display.getDisplay(this).setCurrent(mainForm);
 }
    }

    private Form mainForm = new Form("Text Editor");
    private TextField m_DocumentName =
      new TextField("Document Name:", "foo", 100, TextField.ANY);
    private TextBox m_DocumentText =
      new TextBox("", "", 1000, TextField.ANY);
    private Command m_GetCommand =
      new Command("Load", Command.SCREEN, 1);
    private Command m_SaveCommand =
      new Command("Save", Command.SCREEN, 1);
    private Command m_CancelCommand =
      new Command("Cancel", Command.SCREEN, 1);
    private TextEditorServiceProxy m_TEProxy =
      new TextEditorServiceProxy();
}
```

Download the Code!
www.JavaDevelopersJournal.com

Aquarius Solutions

www.aquariussolutions.com

Northwoods Software Corporation

www.nwoods.com

InetSoft Technology Corp.

www.inetsoft.com/jdj

Architecting Mobile/Wireless

Your organization has decided to extend your enterprise applications into the mobile and wireless space. Congratulations! It will be an exciting time in your software development shop and your customers will no doubt be waiting with great anticipation for the results.

Written by James White

When I hear that organizations have decided to enter the wireless/mobile space I often think of the television commercial in which two businessmen say confidently to each other, "We can do that!" And then in the next breath they mumble, "How are we going to do that?"

If your organization has decided to enter the Java mobile and wireless arena, but you aren't sure what you're getting into, then stay tuned, because this article is for you. It discusses the two general areas to consider when entering the Java mobile and wireless technology space, which is primarily guided by the Java 2 Micro Edition (J2ME) specifications.

First, I discuss the issues involved in collecting the requirements and dealing with the user community as you plan the development of your applications. You're probably familiar with how to gather requirements and design a system with your user community when the target platforms are desktops and servers. I'll provide some real-world experiences and tips on working with your user community to scope out and design mobile and wireless applications.

Second, we examine some of the software engineering issues that go into making your application. These include issues in dealing with a limited user interface, security, data synchronization, performance, portability, application support, and configuration management. You'll encounter several if not all of these issues when you start to develop your application. At least you should know where some of the design and development "alligators" reside and have some tips on how to deal with them.

Dealing with the User Community

Java mobile and wireless applications will often work inside resource-constrained devices like cell phones and PDAs. In many cases, your user community will want to perform many of the same types of operations and communicate using the same data as when they're operating their desktop computers. The trick is fitting that functionality and data into a small device. How do you do this? By helping your customers understand this new paradigm, and allowing them to help you "shrink" the application and data into the smallest footprint possible.

Pramati Technologies

www.pramati.com

Constraint Busters

Near the front of every J2ME specification is information about the types of devices that the development environment and the tools built on the specifications are meant to support. Typically, this is a list of constraints on the computing device that the J2ME specification has been specifically designed to address. The list usually includes constraints on one or more of the following:

- Memory
- Power
- Connectivity
- User interface

These will be the most frequently incurred obstacles in the development of your J2ME applications. As you prepare to design and build your application, push to find out any and all information that may challenge these constraints. Continue to ask yourself and the user community "how" types of questions.

- How much data will actually be used to complete an operation and what data is nice to have?
- How long will the application be used on a given day?
- How often will the system be recharged?
- How long between data synchronizations?
- How convenient will it be to enter the data by keypad or Graffiti?
- How focused will the user's attention be on the device/application?

"Once your application is fielded, more knowledgeable users will make your life infinitely easier as issues begin to manifest (as they always do) in the field"

Setting Expectations

The user community, in the span of their collective lifetime, has seen incredible advances in information systems. They now rely on laptops and desktops connected to company networks and servers for a wave of information. However, porting what might be on a desktop or server onto a Palm device will be tough.

Your first task as a new mobile/wireless architect is to get in touch with and hopefully manipulate the expectations of your user community. Early and often, you must communicate what is possible and what may be feasible but yet inappropriate for mobile/wireless technology. The key is to get the users out of the mode of expecting the next, better laptop or desktop. They've become accustomed to seeing more powerful systems and resources made available with each new platform or release of your software. At least as far as the hardware is concerned, moving to a truly mobile or wireless device will be a big step back for them. Start working on this understanding early.

Good/Bad/Ugly Mobile and Wireless Applications

J2ME mobile and wireless technology is a tool and tools can be used appropriately or inappropriately. J2ME can be used very effectively to extend enterprise information systems out to the mobile and wireless users in need of corporate data. For example, J2ME can be used effectively to put pricing calculators and updatable product catalogs in the hands of sales force personnel.

A J2ME application can also be used to coordinate the schedules and maneuvers of company personnel and equipment such as those owned by a delivery service. J2ME can provide a good platform when you need to have the client device perform an operation in both connected (usually wireless) and disconnected modes.

But J2ME has its limits. Using J2ME on a cell phone to take an entire bank loan application may be feasible, but probably not very useful. How often will a loan officer make a house call, and how easy would it be to enter all that data on a cell phone keypad? Therefore, don't just look at whether moving an enterprise application to mobile/wireless platforms is possible, look at whether it makes sense. In other words, what is the value of having the application on a mobile/wireless platform?

Device Familiarity

How many of your users currently have wireless/mobile devices? If they've never used a Palm device or only use their cell phones to make telephone calls, then it's probably a good idea to educate them to the paradigm. Operating an application running on the Palm OS or using a cell phone keypad to enter character data is different than running a Windows-based application or using a Web browser on a laptop or desktop system.

Get the targeted devices for your applications in the hands of your users early. They won't contain your application yet, but the paradigm familiarization will help you in two ways. One, users of these devices will have a better idea of how the applications on these small devices work. They'll be better equipped to help you create requirements and designs for your application. Their expectations will be easier to manage when they know more about the device and how it typically operates.

Two, once your application is fielded, more knowledgeable users will make your life infinitely easier as issues begin to manifest (as they always do) in the field. If your user community can't operate and help maintain their own devices, you'll have a much tougher time getting important diagnostic information from them.

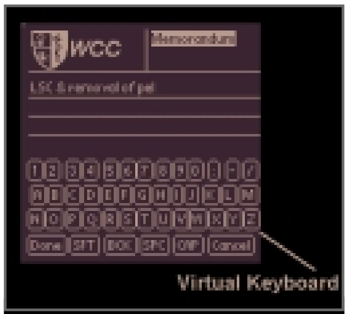
Remember, these devices are mobile. The users will have to get software updates from you and learn how to do maintenance and support activities. The more comfortable the user community is with the device, the less time the devices spend in FedEx packages between the user community and you, and the more time they spend on the road where they provide value to the company. Give them the devices and allow them to play with the address books, calendar software, and other software already present. The more they get attached to the device, the more likely they'll want to see your application succeed.

Requirements Gathering and Design

Whether you develop formal requirements and analysis documents, or informally get your users together to determine what functionality should be in your application, make every effort to separate yourself from the lexicon and paradigm of today's desktop or laptop applications. Terms such as *file*, *mouse*, *dropdown box*, *menu*, or *window* may have no real equivalent in J2ME platform(s) and applications. For example, many phone and PDA devices have an operating system that

Softwired, Inc.

www.softwired-inc.com


WCC
Memorandum
LSC & removal of pel
1 2 3 4 5 6 7 8 9 0 . - /
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
Done SFT BCK SPC CAP Cancel
Virtual Keyboard
FIGURE 1 Virtual keyboard

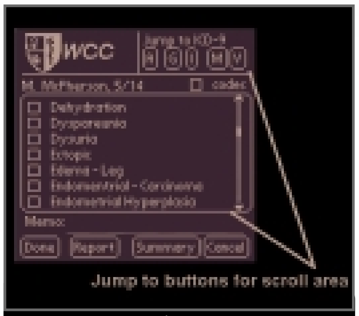


FIGURE 2 Jump-to buttons

has no real file system, much less a directory or file.

Start the paradigm shift right away. Don't let terms and patterns of bigger and more powerful platforms creep into your requirements and design discussions. Expectations can again be kept in check and you won't be left with the task of how to implement or mimic desktop-style functionality on a resource-constrained device.

As you gather your requirements and design your application, also consider where, not just how, your application will be used. A small device can be transported into many environments in which your applications may not be today. For example, your J2ME application may be operated by a salesperson who is seated in front of a client, or in the hands of a truck driver whose focus, hopefully, is on the road and not on the device. The device may be used in bright sunshine or in the dark. It can also be easily lost or stolen.

Most of us have been developing applications that reside on desktops and laptops that live in cubicles or offices and are operated, typically, by chairbound users. Do your requirements and application designs encompass not only the desired functionality requested by the user community, but also the surrounding environment that impacts the use of the application? For example, a lot of information in tiny font on a single screen may require more attention than the user can spare at any one time. Likewise, sales force applications that beep or make other sounds may offend prospective clients.

Explain the Limits

As you begin to understand the requirements and work with the user community to design the J2ME application, know the limitations of your target platforms and share this information with the users. If, for example, the user community wants to put a product catalog database and an application for accessing it on a PDA and you know this database is 2M larger than all the storage capacity of the biggest PDA, then share this with them. I like to tell architects to "do the math" with their users. You don't have to be a computer hardware engineer to understand that 4M of data will not fit into a system capable of holding 2M of information. They will get it and help you find ways to work within the limitations when they know what they are.

For example, in a recent project, a group of doctors helped whittle down an enormous database of clinical diagnoses used in insurance billing activities to the few hundred most often used ones. Thus they could use this application to complete more than 90% of their billings, an acceptable percentage to this group of doctors.

When you're not sure of the performance or capabilities of your device, do some quick prototyping. Wireless transmissions in connection with data synchronization are often areas that require some testing to determine if the transfer rates and processing times are acceptable. If the user community is aware of the results, they may even be prepared to put standard operating procedures in place that go into effect with the delivery of your application and the devices. For example, in another recent project, frequent (sometimes as many as three times a day) synchronization of data was required by the user in order to avoid a very time-consuming and difficult data synchronization that occurred at the end of each day. This was made part of the user documentation and enforced by the product manager.

Dealing with the Devices

Once you understand what mobile and wireless functionality you want to deploy to, your task as an architect or developer is to design and build the application. A few items that impact success and plague new mobile/wireless engineers are listed here.

Portability

The benefit of using J2ME or any micro Java environment to build your applications for mobile and wireless devices is writing software that's portable to a variety of platforms. Unlike the Java 2 Standard Edition or Java 2 Enterprise Edition, J2ME application portability requires more architectural foresightedness. A J2ME application is built on a configuration that provides the basic Java services for a large number of devices and a profile that provides for the specific needs (such as graphical user interfaces, persistent storage) of a specific set of devices such as cellular telephones and pagers. Portability across platforms served by the same configuration/profile combination is straightforward. Portability across configurations and/or profiles can be more complicated.

The user interface creates the most issues in application portability. Some of the issues surrounding user interface portability are discussed later in this article. A good measure to take, so that your applications port more easily, is to partition the application along UI and business logic boundaries. Thus the model-view-controller paradigm applies to your J2ME application as it did in your J2SE or J2EE applications.

In general, architects are left with two general approaches to dealing with issues of portability. At one extreme, the architect can choose to isolate and modularize the piece of the application that may vary across platforms. Partitioning out the UI exemplifies this type of development. Layering the application, however, can lead to more code that can cause footprint issues.

At the other extreme, the architect can choose to utilize the lowest common denominator available among the configuration/profile combinations used on the various platforms. For example, floating point numbers are not supported in the Connected Limited Device Configuration (CLDC) but are supported in the Connected Device Configuration (CDC). To avoid any potential porting issues between these two configurations, an architect taking the lowest common denominator approach would simply avoid using floating-point numbers in an application at the expense of greater functionality in larger platforms.

User Input

Given the lack of a keyboard and mouse, data entry on the small screen of many J2ME devices is a real challenge. Even though many devices now have foldout keyboards for use with their device, this type of apparatus can make the device less mobile. In most cases, user input will be handled with a pointing device like the stylus on a PDA or via a small keypad such as that found on a cell phone or pager. Many PDA devices, like Palm OS devices, offer a mechanism for capturing stylus movement over a touch-sensitive area of the display. Even these input devices can create user backlash because of the need to learn special shorthand for data entry and the difficul-

Fiorano Software

www.fiorano.com/tifosi/freedownload.htm

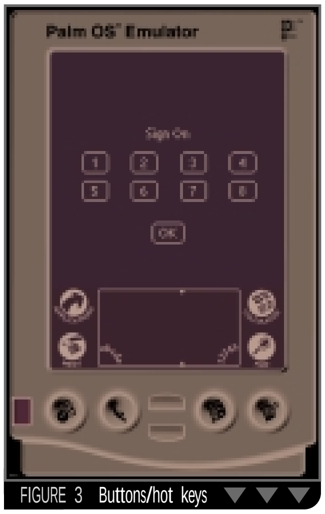


FIGURE 3 Buttons/hot keys

ty the system sometimes has in interpreting some users' input.

When designing user interfaces, I encourage developers to think maximum "pokeability." In other words, attempt to create applications that can accomplish 80% or more of their operations through the touch of a single key/button or the "tap" or touch of the stylus to the screen. Where text entry is required, at least offer a virtual keyboard to allow the user to tap in characters from a keyboard-looking display shown on the screen (see Figure 1). Many of the PDAs offer this capability built into the device, but the Java environment may or may not take advantage of the feature.

Scrolling is another chore that can be a negative experience in small devices. Given the small screens, it's difficult to design user interfaces that don't require some form of scrolling. However, trying to manipulate a very small scroll bar on a small screen can be an exercise in hand–eye coordination. Horizontal scrolling should be avoided at all costs. A limited amount of vertical scrolling is tolerable, but you may want to provide "jump-to" buttons that allow for quick and easy movement in a scrolled area (see Figure 2).

In a similar light, users can easily get lost inside a J2ME application unless some forethought is given to navigational concerns. Unless you custom-build it, there are no menu bars or forward/back buttons as there are in Swing applications or Web browsers. In an application with multiple "screens," think about providing buttons/hot keys to switch to the various displays. Wherever possible, try to avoid having the user remember any data, or worse, having to compare data across screens.

User Interface

J2ME is a set of specifications for devices ranging, noninclusively, from the smart card to the laptop computer. The type and shape of user interfaces in this range of computing devices is vast. Creating the user interface is tough enough – making it portable is even tougher. Several factors should be considered when creating J2ME graphical user interfaces.

The screen size, for instance, will vary among platforms. The typical cell phone may have a screen size no larger than 9,654 pixels. The screen size of Palm OS PDA devices is 160x160 pixels. Most of the J2ME specifications don't include a layout manager to help you cope with these screen-size differences. Therefore, your code will have to serve as a layout manager. There are getWidth() and getHeight() methods that can help your code obtain the screen dimensions, but some work will be required to arrange information and UI widgets on the small screen, especially if seamless portability is required.

Also, many of today's small platform manufacturers are introducing color screens. Until the use of color in these small devices is ubiquitous, avoid the use of color in your UI. In systems that don't have color, it may be very difficult to differentiate some elements of your user interface on a screen that must attempt to gray-shade your use of color. Furthermore, many of the systems still have passive matrix displays, which can make reading some colors on the screen extremely difficult in any kind of natural light.

Finally, unlike the Swing package that universalized Java UIs on platforms that run standard Java, the widget set among the various J2ME specs and other micro Java environments vary. As suggested above, if portability is a chief concern, you may wish to seek the lowest common denominator in widget availability across the various specifications. If you wish to take advantage of more sophisticated UI widgets in higher-end devices, you'll want to isolate the user interface as a separate layer in your application so that swapping different UIs based on system capabilities does not create a wholesale application rewrite problem.

Performance

Depending on the implementation of the J2ME specification, the virtual machine may perform at one-third the speed of the standard JVMs. Given the constrained resources of the device, performance will always be a concern. Here are some performance tips to consider while building your application.

Due to resource constraints, garbage collection algorithms inside many J2ME virtual machines are not as high-performing or sophisticated as those found in larger JVMs. Therefore, some performance increase can be obtained by avoiding garbage generation. A few ideas to help with this goal:

- Use StringBuffer for mutable strings.
- Pool reusable instances of objects like DateFormat.
- Use System.gc() to jump-start or push the garbage collection process.

Once you've determined that the application is working properly, compile the code with debugging information turned off. This not only increases the performance of the application, but also reduces its footprint. To compile your code with debugging information turned off, use the -g:none switch in your compile command.

Where possible, avoid deep hierarchies in your class structure to further improve your performance. Deep hierarchies require the virtual machine to load and link the superclasses.

Many third-party virtual machines generally perform many times faster than Sun's reference implementation. The options may be limited depending on your platform needs, but at least explore and consider purchasing a third-party VM.

Data Synchronization

Data synchronization deals with the management of data on different systems as if it were all on one system. In mobile/wireless applications, data will be sent to and potentially changed on a device where it must then be returned and reintegrated (synchronized) with the rest of the enterprise data. In the last few applications I've written, this functionality has made up as much as one-third of the codebase.

The first issue that must be tackled with regard to data synchronization is to determine which format will be used to exchange data between the devices and the server. XML would seem the logical choice, but it carries a price. XML impacts the size of the data transmitted and XML parsers may be too much for the allowable footprint. Small third-party parsers are now available but they'll still add to the footprint. Serialization is not available in every J2ME environment, so in many cases you may be left with simple delimited data as the means to send data.

Given an understanding of the data and how it will be transmitted between device and server, there are a number of distributed data questions that J2ME architects must resolve. Will data be locked on the server while it's used on the device? How much time does the client device have to modify the data before the server must reclaim ownership and allow others access to the data? If data is not locked, how many clients will have access to the data and what happens if both try to update it? Issues in data use and synchronization can be incredibly complex and require some extra design time.

Help is on the way. Several database vendors are offering

Sitraka

www.sitraka.com/jclass/jdj

"micro" versions of their database that not only manage persistent storage of the data, but also help perform data synchronization between the enterprise and mobile databases. However, these databases may not be available for your target platforms or may require too many resources to satisfy your needs. Also worth investigating is SyncML. SyncML (www.syncml.org) is a developing open-industry initiative for developing standards for data synchronization in both wired and wireless networks. Until third-party efforts and standards such as these are available and small enough to satisfy requirements, data synchronization will continue to occupy a large amount of design and development time.

Security Needs

In most cases, when using off-the-shelf mobile/wireless hardware, your application will utilize standard cellular telephone communication channels. These channels are open to eavesdropping and therefore theft of information. Encryption packages, such as those provided by the Legion of the Bouncy Castle (www.bouncycastle.org), can give your application 128-bit encryption if necessary – especially since many J2ME environments don't yet support SSL.

When considering the need for security, consider the possibility of loss or theft of the device. Compared to a desktop or even laptop, misplacing or stealing a cell phone or PDA is extremely easy. Therefore, you may want to consider whether the information on the device is password protected. It has been my experience that many users resist having to enter some authenticating password in order to use the applications and data on a device they soon come to view as theirs. However, these devices may contain very sensitive or important information (such as patient information or customer lists) with the wireless capability to access your enterprise systems. At least identify and analyze this risk in your requirements sessions.

Signing on may prevent unauthorized access to data in the event of loss or theft (see Figure 3).

Field Support

As your development efforts wind down, the real work in fielding and supporting your mobile/wireless application will begin. Remember, once you prepare and distribute the devices, you may never see the device again. On many of the projects that I've worked on, the mobile users were located throughout the U.S. Once they get accustomed to having the tool, they'll be reluctant to part with it to enable you to add or remove software or data. Designing and planning how to deal with deployment and maintenance issues once the system is in the field cannot be overemphasized. Some consideration and solutions should be found for the issues covered here. Experience has taught me that support has been the most difficult part of building successful systems, and is often the most overlooked part of development.

How will you provide software updates? Even the best software will have a few bugs or changes, especially in early releases. How will these be distributed to and installed on the devices? How does reference data get updated?

Reference data is data that typically doesn't change that often, but is used in running your application. Examples of reference data include lists of states and zip or area codes. Your application may include a product catalog or a list of plant locations as reference data. If this changes, how will this be updated on the devices? Data synchronization, as suggested earlier, can be used, but this may require more time and/or bandwidth than is reasonable for large sets of reference data.

How are backups of the device handled? Hopefully the users have been well trained and well disciplined enough to perform their own backups. Unfortunately, most users perform backups only after experiencing a failure that teaches them a lesson. How do catastrophic device failures get handled? Hardware does fail. If the user is able to get the hardware repaired or functioning again, will they be able to reload the applications and data on the system? If not, how will this be accomplished?

Whenever possible, build your applications in such a way that most of these issues can be handled automatically once the device synchronizes itself with your enterprise systems. This requires more work and more code on your part. Often your application must handshake with a server and both must determine what is on the device and what must be updated. More code means a larger footprint. J2ME specifications have been written to deal with some automatic updating, especially in the area of application updates through the Java Application Manager. However, J2ME is not yet equipped to deal with all types of updates.

"Designing and planning how to deal with deployment and maintenance issues once the system is in the field cannot be overemphasized"

Some of these synchronization operations will require relatively stable and good communication networks. Slow (9600bps) wireless network speeds that drop after a couple of minutes may not allow for wireless and automatic updates, backups, and complex synchronizations. Prototyping here will help you understand these types of limitations. Well-trained users, no matter what the actual mechanism for field support, will always make your job easier and provide you with alternatives when even the best-laid plans fall short in a critical event. If the users can actually help load software and data on their devices, you have a backup field support plan.

Summary

Extending your enterprise applications onto mobile and wireless devices using J2ME can be of real value to your organization. But, as we have seen, there's nothing small about building applications for cell phones, pagers, PDAs, and other such devices. In fact, there may be more issues and factors to consider as an architect of J2ME applications. As with most software development, understand the issues you may encounter and start working on them early and often.

Author Bio

James White is wireless practice manager and senior consultant for Fourth Generation, Inc., based in St. Paul, MN. He is the coauthor of Java 2 Micro Edition *from Manning Publications and has written several articles on Java and object-oriented topics. Jim also speaks frequently at industry conferences and spoke on J2ME at this year's JavaOne.*

jwhite@fourthgen.com

Infragistics, Inc.

www.infragistics.com

J2ME J2SE J2EE Home Labs

AshnaMQ 2.0 Standard Edition

by Ashnasoft

REVIEWED BY ANDREW WU andy@n-ray.com

info

Ashnasoft Corporation
39111 Paseo Padre Pkwy, Suite 213
Fremont, CA 94538
Web: www.ashnasoft.com
Phone: 510 792-6335
Fax: 510 792-6351
E-mail: info@ashnasoft.com

System Requirements
Platforms: Windows NT/2000, Solaris 2.6, 7, and 8, Linux (Red Hat 6.2 or later)
Server: Runs best on JDK 1.3.1
Regular clients: Can run on JDK 1.2.2 and higher
Mobile clients: Runs on J2ME Wireless Toolkit 1.0.3

Test Environment
Platform: Windows 2000
Processor: Viglen PII 700
Memory: 512MB

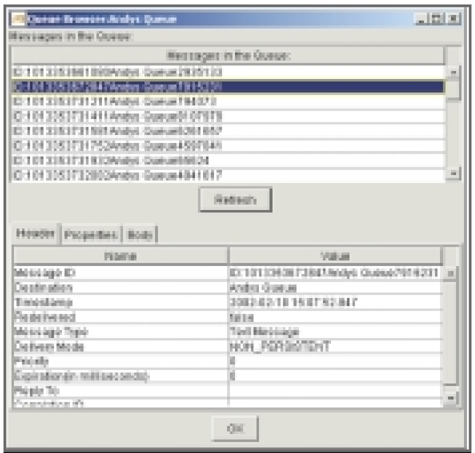
Since its release in late 1997, the JMS API has established itself as a core component in the Java Enterprise suite. A number of vendors are offering implementations of JMS with one of the latest coming from Ashnasoft Corporation. Although new to JMS, Ashnasoft has brought its experience and a reputation from JTurbo (sold to New Atlanta last year) to the scene.

First released in November 2001, the current version (released in February) is AshnaMQ 2.0 Standard Edition. AshnaMQ is a JMS 1.0.2 implementation written purely in Java. It's a product that Ashnasoft claims offers better performance and higher reliability than their competitors. The performance and scalability benchmarks and test harness are available from the Ashnasoft Web site. It supports the full range of Java editions – J2SE, J2EE, and J2ME – with good support for mobile messaging solutions. A server-based implementation, it offers everything you'd expect from a JMS implementation, and a bit more.

Installation of AshnaMQ is painless, to say the least. I had it installed and was running one of the sample programs before you could say "Java Message Service." Well, perhaps a slight exaggeration there. But it really is just a case of starting the JMS server, executing the appropriate provided script to set up the environment, and running one of the examples. There are a host of well-commented examples to cover all the core messaging options, and there are also examples of integration with servlets, JSPs, and EJBs, plus a midlet example.

Two of the more tangible features of AshnaMQ that impressed me were the administration GUI and bundled documentation. The administration tool certainly makes AshnaMQ "easy to administer" (as it says on the front of the box). There's also a command-line interface for the power user in you, but the administration GUI was my preferred option. It enables you to easily create/delete topics and queues on the server, and also control the security aspect of messaging through setting up user accounts and access control lists.

The main highlight for me was the "development tool" feature of the admin GUI. It allows you to explore/play around. From the screen shown in Figure 1, I was able to dynamically create connections, sessions, and, subsequently, multiple senders, receivers, and browsers of messages for sending/receiving/examining messages. Also, a facility to monitor all connections, topics, and queues is provided. For example, you can see how many senders/receivers there are for a particular queue.



Messages in the Queue:

Refresh

Header | Properties | Body

Name	Value
Message ID	[illegible]
Destination	[illegible]
Timestamp	[illegible]
Redelivered	[illegible]
Message Type	[illegible]
Delivery Mode	NON_PERSISTENT
Priority	[illegible]
Expiration (in milliseconds)	[illegible]
Reply To	

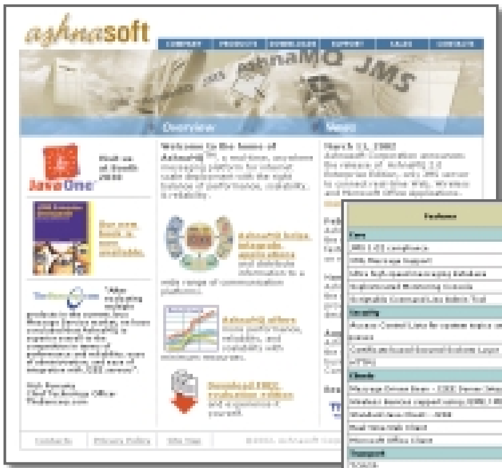
OK

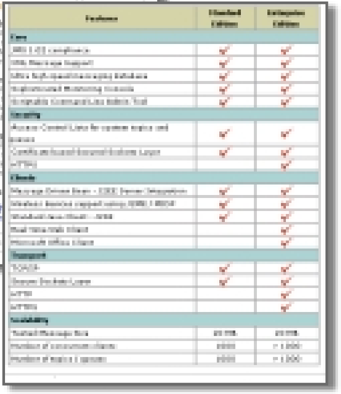
FIGURE 1 AshnaMQ Administrator

The documentation included with the installation provides quality manuals for the different aspects of using AshnaMQ, from the administration side of the fence to the programming. It's a rich source of examples that includes guidance for integrating with EJB/servlet/JSP engines.

Summary

AshnaMQ is good package. It may be lacking the maturity of some of its competitors who have been around for years, but it does claim to be superior in some departments. It offers extensive quality documentation with enough examples to get you started on whatever aspect of the JMS package you want to use. It's so easy to install and use through the admin GUI that it's well worth downloading the free Developers Edition and giving it a bash.





Features	Standard Edition	Enterprise Edition
[illegible]		
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]		
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]		✓
[illegible]		
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]		✓
[illegible]		✓
[illegible]		
[illegible]	✓	✓
[illegible]	✓	✓
[illegible]		✓
[illegible]		✓
[illegible]		
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

Ashnasoft Corporation

www.ashnasoft.com

WebLogic Portal 4.0

by BEA

REVIEWED BY JOHN WALKER *walk@sbcglobal.net*

The appeal of Web portal applications relies on how conveniently they provide a coherent and unified gateway to dynamic content and applications that are tailored to users' needs and interests. The goal of portal applications for the enterprise is to attract users with specialized content and services, and then retain these users by personalizing their experiences. To accomplish this, expertise in a variety of areas is required from different groups that support the application.

Java programmers can create components using a variety of technologies to implement the content and services users will be provided with, while business analysts decide how these components should be used and accessed by users. The issue for developers is to find a framework that allows them to work with implementers as seamlessly as possible. Developers should be searching for the best technologies to build the components that support the Web site's architecture, while the business analysts figure out how these technologies can be used to support the business objectives of the enterprise.

Happily for both developers and implementers, WebLogic Portal 4.0 provides the framework that allows both groups to coexist harmoniously. Based on the award-winning WebLogic Server, and positioned as the cornerstone of the WebLogic Enterprise Platform, the WebLogic Portal 4.0 is a framework that provides the full functional feature set developers expect from a maturing third-generation portal product.

Beginning with the product's Portal Foundation Services, HTML and JSP developers can

info

BEA Systems, Inc.
2315 North First Street
San Jose, CA 95131
Web: www.bea.com
Phone: 800 817-4BEA
Fax: 408 570-8901

Test Environment
Computer: Dell Precision 340 Workstation
Processor: 1.80GHz Intel Pentium IV processor
Hard Drive: 40GB Disk
Memory: 512MB RAM
Platform: Windows 2000 w/Service Pack 2

Specifications
Platforms: Any platform with JDK 1.1, 1.2 support
Pricing: $57,000 per CPU, price includes the clustered version of WebLogic 6.1 Application Server.

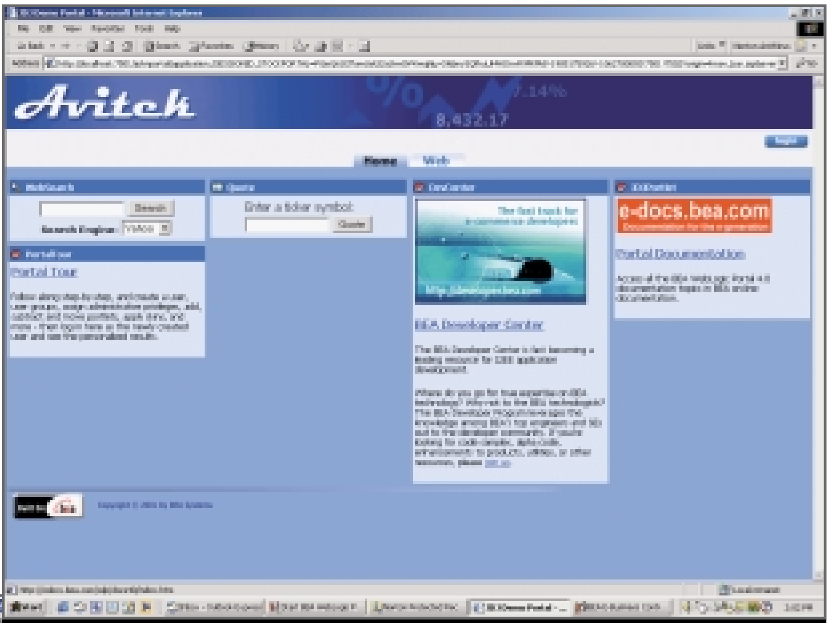


FIGURE 1 Demo portal

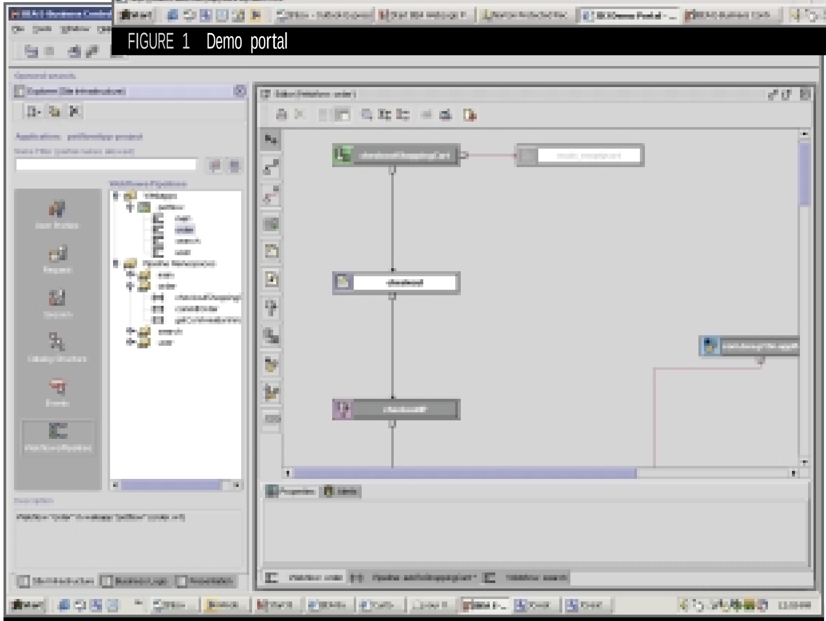


FIGURE 2 E-Business Control Center

Compoze Software

www.compoze.com/jdj

WebLogic Portal 4.0 by BEA

quickly create portal applications that contain portal pages made up of portlets. Portlets can be perceived as individual windows arranged inside portal pages that can be used to obtain access to specialized services or content. The concept of portlets allows for the easy integration of applications specific to the enterprise, whether it's B2B, B2C, or B2E. For example, legacy portals can be integrated with portlets as well as with any existing Web services.

A developer with JSP and Java experience can easily create the underlying JSPs that provide the templates for each portlet's definition. After that, layouts and skins determine the position of the portlets on each page and their look and feel. The intelligent design of the development framework abstracts the components away from their implementation layer. Individuals concerned with the implementation of the business plan can use the JSPs built by the Java developers and the skins and layouts provided by the HTML developers to easily organize and construct the portals on their own.

Installation and Configuration

Before installing WebLogic Portal 4.0, WebLogic Server 6.1 should first be installed along with Service Pack 1. Make sure the samples for WebLogic Server 6.1 are also installed; if they're not, WebLogic Portal 4.0 won't start. A Cloudscape database is provided along with the samples and it's this database that WebLogic Portal 4.0 relies on. The installation set for WebLogic Portal 4.0 was obtained from http://commerce.bea.com/downloads/commerce_servers.jsp. Be sure to consult the installation guide before proceeding. It will inform you of the necessary file sets needed to run the product and the order in which they are to be installed.

If the product is being downloaded from the Web site, be sure to download wlportal 400_with_sp1_win.exe, ebcc400_with_sp1_win.exe, and license_wlportal400.bea. The ebcc400_with_sp1_win.exe contains the installation files for the E-Business Control Center. Even though it's in a separate installation set, it's an integral part of the WebLogic Portal 4.0 platform. Once all the necessary files are obtained, the installation guide is very thorough in explaining all the steps to complete the installation. The examples and documentation are excellent.

Working with WebLogic Portal 4.0

The E-Business Control Center is the application used by business analysts and developers to organize and implement the components used to create the features and services that make up a portal application. The sample portal applications are a good starting point for this investigation and easily demonstrate all the key features the framework provides.

Starting with the "Using the E-Business Control Center Portal Tool" documentation, I quickly created, deployed, and tested my first portal. Following the steps provided in the documentation and using the predefined components supplied with the samples, this process was very straightforward. Portals are created from inside the context of a Web portal application, which supplies easy access to all the necessary resources to support and create them. Before proceeding any further it's important to understand which supporting resources are provided (see Figure 1).

When creating the portal, a business analyst need only select the preconfigured components provided by Web designers and Java developers. JSPs are required to render the header, footer, navigation, and content areas of the primary portal page. Next, which skins, layouts, portlets, and user profiles to implement must be determined. Skins are cascading style sheets that contain the look-and-feel parameters for the portal, while the lay-

Apptricity

www.apptricity.com

out determines how the portlets will be laid out on each page. Layouts are simple HTML table layouts: each cell contains a tag that's used to determine the appropriate portlet to place in it when the page is displayed. User profiles contain a list of the attributes describing the potential users of the portal. Profile attributes are associated with users in portal groups who are using a separate application known as the Administration Tools application.

These attributes can be used to determine which content and entitlements will be provided, for instance, whether the user qualifies for a discount on the particular products he or she has added to the shopping cart. Entitlements dictate if the user can take an active role in personalizing the content, layout, and look and feel of the portal. Personalization is more passive and determines what content the user will be exposed to during the course of the session. Using a campaign component I was able to expose a particular banner ad to a segment of users who defined themselves as readers of a technical journal. Most of the effort required to achieve this result was accomplished from inside the E-Business Conrol Center. I only had to update a tag reference in a JSP file to point to my new image file (see Figure 2).

Once all the components have been added, it must be determined how the user will be exposed to them as he or she navigates through the pages. A Webflow is associated with a portal and acts as the navigation framework for the session. Using Webflow, a business analyst can determine which actions a user can choose from on a page or portlet and what data processing will take place as a consequence of that action. This is my favorite feature. Analysts can graphically represent what the users will experience when moving about the portal, from the presentation page they'll visit to the servlets and EJB used to validate and process user data. No need to bother another developer any time a change has to be made to the navigation framework; this would happen only if a feature was added. Webflow is represented graphically as a series of nodes that the control flows in and out of. These nodes can represent the page the control will pass to or a Pipeline object that represents an action to be taken. Pipeline could be an EJB that places or removes a product from a shopping cart.

Summary

WebLogic Portal 4.0 provides an environment that not only allows for the integration of different skillsets, but enables them to work more efficiently. Ready-to-use examples are provided to guide and educate developers and serve as templates for the quick implementation of new features. The E-Business Control Center is not only an environment that allows for the easy creation of these products, but also for their intelligent management and implementation as well.

Product Snapshot

Target Audience: Business engineers, business analysts, Java programmers, and application architects

Level: Beginner to advanced

Pros:
- Feature-rich product
- Plenty component templates and examples
- Easy-to-use graphical Webflow interface

Cons:
- None significant

JDJ Edge

www.sys-con.com

RequisitePro

by Rational Software Corporation

REVIEWED BY SCOTT SILVERMAN scott.silverman@catavo.com

info

Rational Software Corporation
18880 Homestead Rd.
Cupertino, CA 95014
Phone: 800 728-1212
Web: www.rational.com

Test Environment:
Computer: Gateway Solo 9150
Processor: Pentium III 333MHz
Memory:128MB RAM
Platform: Windows NT Workstation 4.0 (Service Pack 6)

Specifications:
Platforms: Windows NT 4.0, XP, 2000, 95, 98, or Millennium Edition
Pricing: Rational RequisitePro Windows Server, Rational RequisiteWeb, and Rational RequisitePro are included in the Rational RequisitePro package for $2,034 for a node-locked license with one year of support.

As IT projects proliferate due to the wide use of the Web for day-to-day business and commerce, it has become clear that the success of these projects can be attributed to proper due diligence. One part of ensuring the success of any IT project is directly related to detailed requirements gathering. Developing solid requirements documentation helps a project's sponsor and development teams come to a consensus on a high-level of "absolute truths" that define the project's criteria for success.

Rational RequisitePro is a tool that helps in the often painstaking task of gathering and documenting requirements. RequisitePro accomplishes this by incorporating an environment in which members of any development team can access and update project goals and requirements.

Overview

In a nutshell, the core of RequisitePro is a tightly integrated database and application that works closely with Microsoft Word. This application enables users to update requirements, either within the application environment or by using the toolbar and menu options in Microsoft Word that RequisitePro installs. RequisitePro offers the following major features:

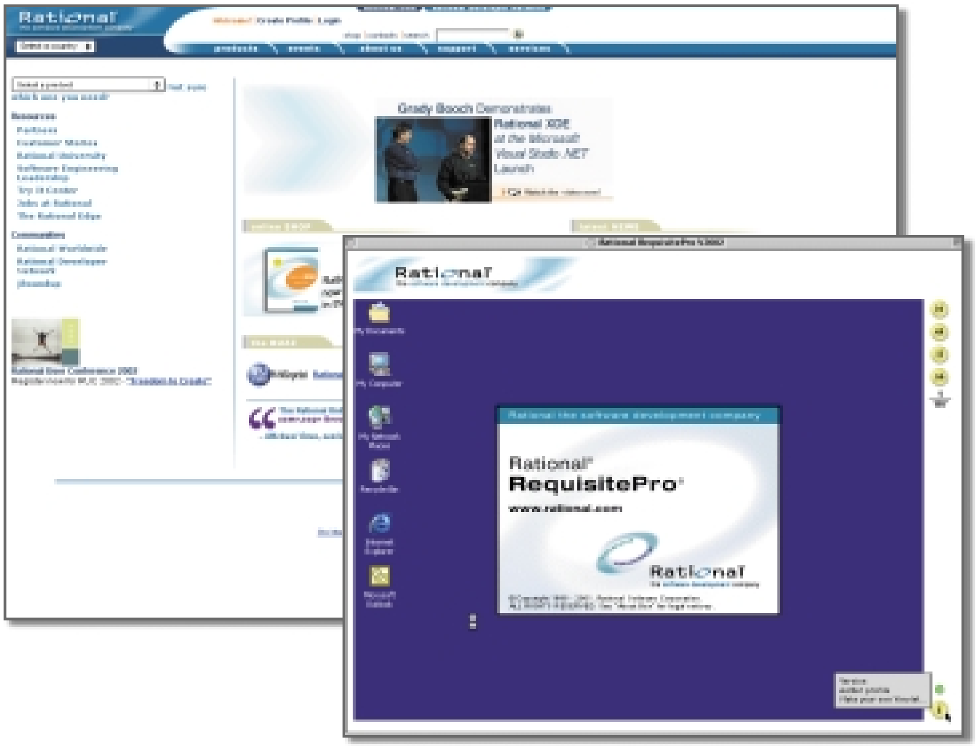
- Full traceability of ever-changing requirements
- Prioritization of requirements
- Templates to identify and manage various types of requirements
- An environment that supports hierarchical relationships among requirements
- A Web-based system for viewing and editing requirements

RequisitePro also enables users to view and manage files via a Web interface. This feature is an important component in a team's ability to access the core rules and specifications governing the development of a system. The RequisitePro license comes with a Windows server and a Web-based component to provide Web access. The product also ships with a Quick Tour as well as a more in-depth tutorial. As a side note, there's full integration between RequisitePro and other Rational development, modeling, and process software, creating a suite of products that covers the full life cycle of software development.

Working with RequisitePro

Installing RequisitePro was relatively easy, although I did receive an error message saying the ODBC driver couldn't be properly configured. This error didn't seem to interfere with the operation of the product and may have been due to a nuance of the specific configuration of my system. I installed the complete product to take advantage of the Quick Tour and the included tutorial. The entire installation required about 100MB+ of disk space.

The RequisitePro window provides an environment that's logical and sensible for the maintenance of requirements. There's a tree view on the left side of the window and the details of the tree are represented on the right (see Figure 1). The tree-view items on the left expand and contract depending on the item



COMING: WIRELESS EDGE CONFERENCE AND EXPO
wireless
BUSINESS&TECHNOLOGY
Lowering Health Care Costs, Out-of-the-Box
IS WORLDWIDE WIRELESS BROADBAND Heading Our Way?
WRITE THRICE, Run Anywhere
WIRELESS Hotels
MOBILE OPERATORS

RequisitePro by Rational Software Corporation

that's clicked. Additions and edits to requirements can be added directly to the proper area in the application window or to the specific Microsoft Word requirements document. When adding items to the Word document, these items automatically update the RequisitePro tree in the application when saved. The windows are resizable, and there's a toolbar and menus to execute the necessary commands.

When using the "traceability matrix" feature, the left side of the menu can represent a grid of the feature's requirements along the top and a list of traced use cases below and to the left (see Figure 2). This was my preferred method to view linked requirements that are dependent on each other. You can also use a "list" type view that has a list of the requirements down the left-hand side, including a numeric reference to the linked requirement. RequisitePro supports hierarchical requirements that allow you to view all parent and child relationships for requirements as well as inherent dependencies.

RequisitePro comes out of the box with some great templates for use case–driven, supplemental, and feature-based requirements. A very important feature is the ability to customize these templates with a document and separate identifier of your own. This allows you to incorporate the RequisitePro product into your own process and methodology.

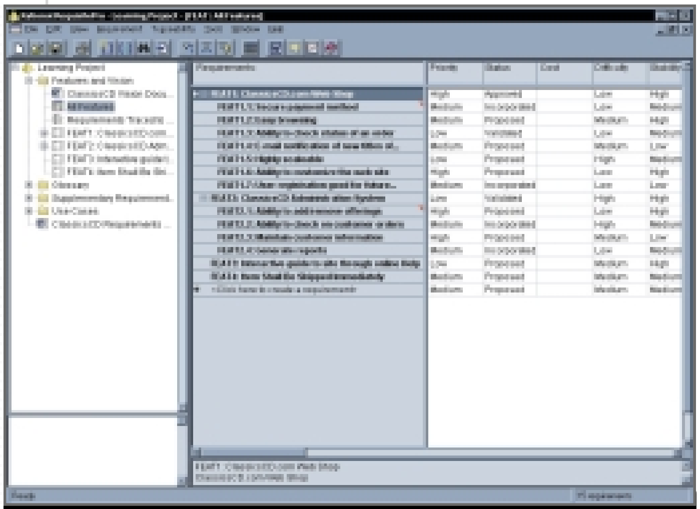


FIGURE 1 RequisitePro window

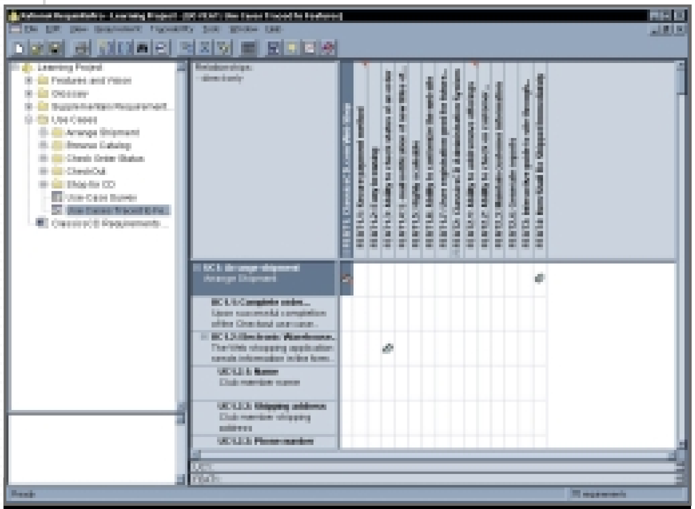


FIGURE 2 Traceability matrix

Simplex Knowledge Company

skc.com

The real benefit of this tool lies in the integration of database-driven requirements tracking and the ability to work in a familiar environment such as Microsoft Word. In addition, project team members can gain access to the documentation and team members can add, edit, or delete requirements, making this product a good tool to add to a mid- to large-scale project.

As for speed, RequisitePro seemed to respond well on my test platform, as long as I didn't try to run too many other applications simultaneously. I usually try to keep open programs to a minimum, as the system I work on is not quite state-of-the-art. I noticed a very minor lag between the update of the RequisitePro window and the Microsoft Word document. With systems above the recommended minimum requirements, I can see how the performance would be excellent.

Summary

Rational RequisitePro does a great job of tracking, editing, and generally making project requirements accessible. The integration with most standard databases and close-to-seamless updating in Microsoft Word makes RequisitePro a helpful addition to the arsenal of project management and business analysis tools. The interface is easy to use and can be learned quickly. The Web-based interface makes sharing valuable project information simple and straightforward. The ability to customize the included templates or create and reference your own is an excellent feature. Any development team working on a mid- to large-scale project will save time and money using RequisitePro's features.

Product Snapshot

Target Audience: Program managers, project managers, business analysts, and project team members

Level: Entry-level business analysis skills and basic to intermediate Microsoft Word skills

Pros:

- Integrated database functionality
- Traceable requirement relationships
- Good tutorial
- Excellent customizable template structure
- Supports hierarchical requirements and access by the entire project team

Cons:

- Requires third-party database program
- May be overkill for some smaller size projects

Dynamic Buyer Incorporated

www.ibm.com/smallbusiness/dynamicbuyer

–continued from page 5

There May Be Trouble Ahead . . .

first ones to complain if I buried my head in the sand and just ignored the threat. We have to look at this together and come up with a strategy that will enable us to effectively take on C#. We'll be getting a lot of heat from all over and we need to be armed with the information and prepared to go back to the drawing board and reeducate the masses. Sadly, they are being led a merry dance by Pied Piper Gates.

Allow me to cite you an example of such blind ignorance and if this doesn't scare you, then I don't know what will. I was recently involved with the Scottish government, discussing technology and what have you, where naturally the topic of Microsoft was high on the agenda. Excusing the fact that these people took a certain pride in believing they knew what was going on and loved name-dropping, the phrase that caught me off guard was the following: "Java? No one is doing that now. Microsoft is no longer supporting it."

Wow! Talk about a major miscommunication. And this from someone who controls budgets for the technology sector in Scotland. Ironically, I believe he really thinks he has his finger on the pulse of technology. It's sheer ignorance like this that scares me the most. Microsoft has successfully planted and nurtured the seed in people's heads that just because it isn't supporting Java in Windows XP, Java is dead. I have to admit I was taken aback and quite flabbergasted when I heard that retort. I really didn't know where to go with that. So much background information was obviously missing that I wasn't too sure if I would come over as patronizing and whether, ultimately, they would understand.

Sadly, this is not an isolated incident. Ever since I started writing about this topic in my editorials, I've been hearing stories from you regarding similar misconceptions and it scares me. We have a beautiful language here in Java; it has achieved industry-wide support and is pushing forward with great velocity. What can we do to support it?

You do realize we need an anthem. All great causes have an anthem. Something for us to get behind and sing!!! Suggestions gratefully received. We need a Java song!

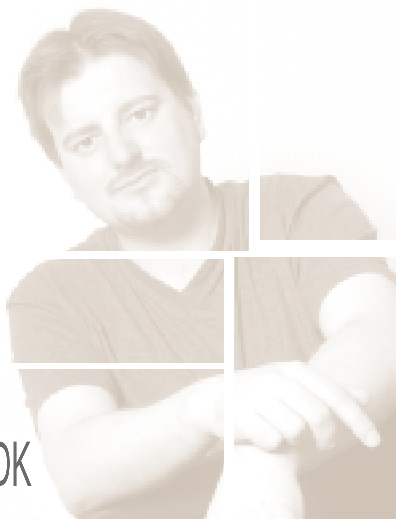
Until next month...

Java Developer’s Journal

www.sys-con.com

AN INTERVIEW WITH SHERMAN DICKMAN BY ALAN WILLIAMSON

JDK 1.4



JDJ Readers Question Sun Regarding the Latest JDK

Sun recently announced the general availability of JDK 1.4. ***JDJ***'s editor-in-chief, Alan Williamson, had the opportunity to sit down and talk with Sherman Dickman, senior product manager for the Java platform. Before the meeting Alan invited the readers of ***JDJ*** to put together some questions for Sherman. The following is a transcript of that meeting.

<alan>: What are the main features of this new JDK release?
<sherman>: Some of the features that stand out are 64-bit support, new I/O, XML support, Kerberos single sign-on capabilities, a reengineered Java technology-enabled 2D graphics engine, Java Web Start software, logging, assertions, IPv6, and a lot more. There's a great overview of the new features in the Java 2 Platform, Standard Edition (J2SE) 1.4 on our Web site at http://java.sun.com/j2se/1.4/.

<alan>: How many classes make up this JDK?
<sherman>: There are just over 11,000 classes in J2SE version 1.4. Note that this includes the implementation and not just the APIs, and does not include Java Web Start software.

<alan>: What has been the most requested feature you've addressed in this release?
<sherman>: In terms of the number of requests, "assertions" and "mouse-wheel support" were ranked very high when the feature list was first drafted, in the fall of 1999.

<alan>: The browser world is still catching up with Swing and developers still can't develop Swing apps to run within a browser without invoking a major download. What is Sun doing to encourage the browsers to take the leap and bridge the gap?
<sherman>: Actually, Netscape and Opera provide excellent support for J2SE. In addition, the J2SE platform is bundled with most major operating systems including Linux, Mac OS X, and the Solaris Operating Environment. There's one browser/OS vendor in particular that has yet to ship the J2SE platform, so we're providing improvements to the Java plug-in software that enables users to run browser applets on the J2SE platform. The key developer advantage of these improvements is that existing applets can be run without the need to modify <APPLET> tags in HTML pages.

<alan>: Why was the release of 1.4 delayed? What was causing the major issue?
<sherman>: We originally planned to release the J2SE 1.4 platform at the end of 2001, however, there were a few things that contributed to our February 13 release date. A couple of Java Specification Requests needed to complete the full Java Community Process cycle. We also wanted to ensure that there was sufficient time to address some of the customer feedback we received from our betas, in addition to providing a bit more bake time into the release to satisfy our reliability goals.

<alan>: Why did it take so long to address the Swing performance and underlying graphics redraw problems?
<sherman>: J2SE version 1.2 was the first major release to introduce the new Java 2D package, which serves as the core rendering engine for J2SE. The focus for this first Java 2 release was feature completeness, and later releases were dedicated to tuning the new graphics system. We identified a lot of tuning opportunities, and in the interests of shipping the 1.3 release in a timely fashion, we decided to spread this work across the 1.3 and 1.4 releases. So the performance of Swing for each release has steadily improved, and we anticipate yet another boost in the 1.5 time frame.

<alan>: How's float and double arithmetic performance relative to JDK 1.3?
<sherman>: I'm not aware of special work to improve float/double performance in 1.4, although bounds-check elimination really helps for code that's floating-point intensive. It's probably a bit faster, but not something we track independently.

<alan>: How about a performance comparison for different sections of the JDK versus previous JDKs.
<sherman>: There's a great 1.4 performance guide at http://java.sun.com/j2se/1.4/performance.guide.html, and I'd like to encourage everyone to check it out. It goes into a lot of detail and serves as a great resource for anyone interested in evaluating the performance improvements from 1.3 to 1.4.

<alan>: **What are some of improvements that were done to the compiler (such as branch prediction)?**
<sherman>: There are two HotSpot virtual machine compilers in the J2SE platform – one tuned for clients and the other for servers. Improvements in the client compiler include better local code quality, improved inlining, providing significant speedups to most Java technology-enabled programs, and optimized performance for new I/O. Improvements in the server compiler include a new deterministic inlining mechanism, array bounds check optimization, new loop optimizations, and a substantial effort toward stability.

<alan>: **Does Sun ever plan to remove any APIs that have been deprecated since JDK 1.0/1.1?**
<sherman>: Sun currently does not have plans to remove deprecated APIs.

<alan>: **In general, what are Sun's intentions regarding deprecated APIs?**
<sherman>: Deprecation is advice to developers that some APIs have been replaced with better ones. It doesn't mean the old API is going away. Going forward we intend to make very limited use of additional deprecations.

<alan>: **Why were assertions added as a change to the language instead of an API?**
<sherman>: A library approach would have worked, but it would have been either ugly (requiring an "if" statement for each assertion) or inefficient (evaluating the asserted condition even if assertions were disabled). Faced with these deficiencies, we felt that developers were more likely to prefer the addition of assertions into the language itself.

<alan>: **Why can't they make Swing more lightweight in terms of being able to utilize a single component without having to drag the entire API along?**
<sherman>: Object-oriented systems like Java emphasize reuse through specialization. Component systems like JavaBeans focus on reuse through configurability and delegation. Both of these approaches have a tendency to produce systems with many highly interdependent general-purpose parts and Swing is no exception. One unfortunate side effect of this is that startup time and the working set grow as the number of classes loaded and used grows. We balance this growth by limiting dependencies that cross functional boundaries, sometimes even at the expense of reuse. This is an ongoing process and it's the primary focus of the Swing implementation work that's underway now.

<alan>: **Are there any plans for a Java 2.0?**
<sherman>: By Java 2.0, do you mean a Java 3 platform?
The development goals behind the J2SE platform are to provide consistent and compatible releases of J2SE technology over time. We recently shipped the J2SE 1.4 platform, and are beginning the planning process for version 1.5, which will release approximately 18 months from now.
The shift from the Java 1.x–based platform to the Java 2 Platform, Standard Edition in 1998 was quite substantial, but the platform is maturing quite nicely with our current release model. There are currently no plans for a Java 3 Platform, Standard Edition in the near future.

<alan>: **Which, if any, of the JAX Pack APIs will be moved to the JDK? What is future thinking about the JDK and Web services APIs? JAX Pack forever or will some "sneak" into the JDK?**
<sherman>: Great question! XML parsing and XSLT support were included in the J2SE 1.4 release. Since all J2SE releases now participate in the Java Community Process, it will really be up to the expert group for J2SE version 1.5 to determine which XML APIs should be included as part of the core J2SE platform. In the meantime, additional XML APIs will ship via the Java XML Pack.

"There's one browser/OS vendor in particular that has yet to ship the J2SE platform, so we're providing improvements to the Java plug-in software that enables users to run browser applets on the J2SE platform"

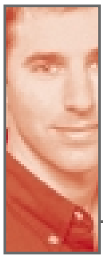
<alan>: **Is the XP plug-in part of JRE 1.4?**
<sherman>: Yes, it is, and will be a part of all J2SE technology releases moving forward.

<alan>: **When they started working on the regular expressions package for 1.4, did they look at other regular expressions packages? Apache has the ORO package, a very nice package, in my opinion.**
<sherman>: Yes, we evaluated other regular expressions packages to get a feel for what functionality a good regex package should offer. The goal was go with a syntax similar to the most popular regex language out there, Perl. Our design profited from the efforts of the ORO project, as well as other earlier efforts.

<alan>: **When will Sun be updating the community edition of Forte for JDK 1.4?**
<sherman>: Today! The Forte for Java Community Edition IDE can be downloaded with J2SE 1.4 from our Web site at http://java.sun.com/j2se/1.4/download.html.

<alan>: **With respect to 1.5, is there anything that should have been part of 1.4 that has now been pushed back to Tiger (1.5)?**
<sherman>: Yes. In the interests of shipping J2SE 1.4 on time with great performance and reliability, we had to make some tough decisions and defer a few features to the 1.5 release. This included the JPDA back-end expression evaluation, a character converter generator tool, concise array literals, socket factory support, and a few features from new I/O, including scanning and formatting and an improved file system interface.

<alan>: **Sherman, on behalf of our readers and myself, I would like to thank you for taking the time to answer our questions. We'll be back when you announce 1.5!**



Missed an issue?

We've got 'em all for you on CD!

JAVA DEVELOPER'S JOURNAL

The most complete library of exclusive *JDJ* articles on one CD!

Check out over 500 articles covering topics such as...
Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, Libraries, Embedded Java, XML, Wireless, IDEs, and much more!

JDJ The Complete Works
~~Reg $119.99~~
Buy Online
Only $71.99

WebServices / XML JOURNAL

The most complete library of exclusive *WSJ* & *XML-J* articles on one CD!

Check out over 400 articles covering topics such as...
XML in Transit, XML Industry Insider, UML, Integration, WSDL, VoiceXML, XML & E-Commerce, Beginning Web Services, Web Services Tips & Techniques, Ubiquitous Computing, Information Management, XML Script, UDDI, .NET, Objects & XML, XML Middleware, and much more!

Web Services Resource CD
~~Reg $119~~
Buy Online
Only $79

COLDFUSION Developer's Journal

The most complete library of exclusive *CFDJ* articles!

Check out over 250 articles covering topics such as...
Custom Tags, ColdFusion and Java, Finding a Web Host, Conference Reports, Server Stability, Site Performance, SYS-CON Radio, ColdFusion Tips and Techniques, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Application Frameworks, Error Handling, and more!

CFDJ The Complete Works
~~Reg $79.99~~
Buy Online
Only $71.99

CF Advisor

The most complete library of exclusive *CFA* articles!

Check out over 200 articles covering topics such as...
E-Commerce, Interviews, Custom Tags, Fusebox, Editorials, Databases, News, CF & Java, CFBasics, Reviews, Scalability, Enterprise CF, CF Applications, CF Tips & Techniques, Programming Techniques, Forms, Object-Oriented CF, WDDX, Upgrading CF, Programming Tips, Wireless, Verity, Source Code, and more!

CFA The Complete Works
~~Reg $79.99~~
Buy Online
Only $71.99

SPECIAL OFFER:
Buy CFDJ & CFA The Complete Works
For Only $129.99

JDJStore.com

Order Online and Save 10% or More!

www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

COLLECT ALL 4
FOR ONLY $229.99
JDJ, WSJ & XML-J, CFDJ, & CFA

‘WE CAN’T LET JAVA GO THE WAY OF WORDPERFECT’

SAYS JAVALOBBY FOUNDER

by JDJ News Desk

MICROSOFT'S $200,000,000 BUDGET FOR .NET *marketing requires immediate remedial action by everyone in Javaland, Rick Ross insists*

He has said it before. He will doubtless say it again. JavaLobby founder Rick Ross is circulating his most urgent rallying cry yet to those who would preserve Java technology in the enterprise and fight off the $200,000,000 marketing campaign that Microsoft pledged to throw behind its .NET Framework, now that it has officially launched Visual Studio .NET. “Make no mistake,” says Ross in a newsletter to JavaLobby members, “this massive campaign is aimed at persuading your peers and managers to choose .NET instead of Java. It’s aimed squarely at you, your job, and the technology platform in which you have invested time and energy to become an expert.”

Ross’s argument is that Java developers “probably have something to lose,” and “may even have a lot to lose.”

“Bill Gates and Steve Ballmer have repeatedly stated that they have bet their whole company on .NET,” Ross warns, “so you can be absolutely sure they have a lot to lose. Expect them to compete ruthlessly, and remember that they’ve established a track record of being willing to play dirty.”

JDJ is not the place to comment on that latter allegation, but looking at the technological side of things, Ross, of course, has a point. When Microsoft launched Visual Studio .NET at VSLive! on February 13, there were simultaneous events around the globe. With its own Java-like OO language, C# (C Sharp), and aimed at the preexisting – and huge – community of somewhere between 6–8 million Visual Basic developers, VS.NET is regarded by many industry pundits as evidence that Microsoft has completely rewritten all the rules of how Windows software is built and deployed.

With corporate use of Windows as an enterprise computing platform already on the rise, the .NET Framework is arguably on course to become pervasive over the next few years, and it’s Ross’s view that Internet technology professionals everywhere will need to know the respective strengths of both platforms in order to advise clients objectively on why J2EE beats .NET for both business and technological reasons.

“We can start by focusing on the fundamentals,” Ross explains. Then, repeating a point he has made again and again in recent months: “most Java developers comprehend instinctively how important it is to have viable alternatives to the offerings of a monopolist.”

> “Internet technology professionals will need to know the respective strengths of both platforms in order to advise clients objectively”

“We understand intuitively,” he adds combatively, “that using .NET leads directly to single vendor lock-in and everything that implies, especially when the vendor is Microsoft. If we can just be clear, articulate, and pleasant while explaining this to people who may be less passionate or knowledgeable than ourselves, then we will be a solid front line of defense against Microsoft’s $200,000,000 campaign to sway public opinion in their favor. It’s a simple beginning, but a powerful one.”

“This is just a beginning,” Ross concludes. “We can’t let Java go the way of WordPerfect!"”

What do you think? How can Sun best make the Java runtime ubiquitous? How can Javaland best resist the Microsoft $200,000,000 marketing push? Is it time now perhaps for Sun to open source the Java implementation, maybe under GPL/SCL? What more should and could Sun do to boost Java? How much is .NET a threat to Java – is it really life or death? Or will the future be interoperable on the Web services model? Can Java and .NET in that case coexist peacefully? To add your comments, go to www.sys-con.com/java/articlenews.cfm?id=1333.

JDJ Feedback Special Report

by Jeremy Geelan

JAVA AND .NET *Now Competing Head-to-Head for the Hearts and Minds of Enterprise Developers. "Large corporations should shy away from .NET" say some; "Why shouldn't developers trust .NET in the enterprise?" ask others.*

JavaLobby founder Rick Ross's cry to Java developers everywhere – "We can't let Java go the way of WordPerfect!" – clearly struck a note with those developers worldwide who share his concern at the impact .NET will or won't have on the continued success of Java in the future, now that Microsoft Corp has begun to spend the massive $200,000,000 budget it has allocated to the marketing of its new framework.

Thoughtful and considered responses to Ross's call to action have been coming to us from developers far and wide. Some are critical not so much of Microsoft as of Sun Microsystems itself, who many feel may be getting its tactics completely wrong.

"If only Sun would loosen its grip on Java standards and certify open source application servers like JBoss," comments Sam Johnston from New South Wales in Australia, "then perhaps we could compete, but so long as we're paying hundreds of thousands of dollars for app and database servers before we even start developing anything, how are we ever meant to get anywhere? A single Enterprise IDE license alone absorbs a large chunk of smaller projects' budgets."

B.J. Schreib feels the same way. "Open it!" he exclaims. "If Java is ever going to have a chance in hell of killing .NET, then Sun has to open up Java. ...by doing so, Sun stands to gain more credibility in the developer community and, as we've seen with Linux, allows (but does *not* assure) the creation of a developer base that can potentially eclipse the speed at which the dinosaur in Redmond can move."

Jonathan Ginter agrees. "If Sun is going to withstand this attack," he says, "perhaps its best strategy would be to finally submit Java as an open source project under strict licensing. I feel that this would merely extend the Java Community Process concept that's already in place. Moreover, it would cripple Microsoft's ability to make an issue of Sun's ownership of Java and strengthen the Java community's claims of vendor lock-in for C#."

"If Sun feels that this would be too much of a free-for-all with little or no quality control," he continues, "then how about finding a way to create joint ownership of the Java standard: a collective approach that could include Sun, IBM, HP, etc?"

Scott Smith can't understand what all the fuss is about. "Why so paranoid?" he asks. "I don't understand the war mentality. Competition is good. I develop exclusively in Java at this time, but three years ago I was a C++ programmer. Three years from now I may be writing exclusively in C#. I'll go where the market demands."

But even Smith thinks opening up Java might help in resisting .NET. "If Sun is so arrogant and short-sighted as to get greedy with Java, they will lose. Sun should make Java open and give open source efforts, like JBoss, certification. The likes of BEA will have to suffer for Java to survive. It's a hard choice, but that's what it comes down to. The least greedy will survive. And, unfortunately, Microsoft can afford to be very 'altruistic' in the short term in order to ensure long-term success." (Java developer Doug Harris loves the idea of Microsoft being altruistic. "Is this like a Scout helping a lady across the street, because it's darker and easier to rob her on the other side?" he asks.)

Mike Wong is more, shall we say, open-minded. Commenting from the Philippines, he says: "Microsoft's strengths lie in the marketing, tech support, and continuous improvement of the products it sells (or bundles). Although I really appreciated and admired the way Java architects created Java, it must...compete. You can't just rely on the beauty of the language in order to rise above the [.NET] challenge."

Finally, and this remains the alpha and omega of Java's continuing success perhaps, Michael Dean reminds us all of Java's tried and true strength, based on his daily, personal experience: "I write/compile/test Java on NT.4 in VAJ, FTP to the S/390 (z/OS) and run it *without change*...that's portability! Where have we ever seen that from MS?"

Amen. And definitely not, we hope, RIP!

The Pros and Cons of Certification

"To Be or Not To Be Certified…" (Vol. 7, issue 2) by Keith Brown is a great editorial.

Finally someone is standing up and saying what the entire community is thinking: certification is just another money-making route for Sun.

Believe me, I've come up against a lot of people who wear their certification badge with pride, but you wouldn't want them anywhere near your project.

Certification is just a memory game, big deal. So if you're certified, well done...be proud of yourself...you've financed Sun a little bit more...you've "bought" an extra line for your CV!!!!

Henry Roswell
runrig10@hotmail.com

I have some credibility problems with Keith Brown. I take his opinions with a grain of salt, since he "has been involved with Java for many years" and yet still needed to take a five-day programming course from Sun, *and* he considers "actually writing code" a "necessary evil" for preparing for this exam. I am studying for this exam to gain a better understanding of Java fundamentals; this process has already benefited me a lot, and I haven't even taken the actual test yet!

Rob Ross
via e-mail

Editor's Note: I believe you've misread the editorial. Keith Brown believes the programmer's exam is a necessary evil, not writing code.

Apples to Oranges

Alan Williamson is indeed correct about the lack of .NET developers on the street ("Scandalous Propaganda," [Vol. 7, issue 1]). It's premature for anyone to insist that .NET will ultimately be faster, especially for the necessary heavy lifting that's done in the middle tiers. Like COM and COM+, .NET is not only CLR, C#, and J#, it's a host of services as well. It's these additional services, like MTS, MSMQ, BixTalk, etc., that will make up solutions based on MS technology...and there ain't no real data yet.

All of this "speed" is attributed to some demos that are really focused on the CLR engine performance. MS is no stranger to strong VMs. In fact, if I'm not mistaken, they had the fastest JVM. So it's little wonder that a .NET application will cruise along just fine. However, their surrogate app server technology (MTS and COM+), while flashy and "easy to use," leaves much to be desired and is not something that will scale up and host millions of objects. As an MCSD and previous user, I speak from lessons learned.

Patrick Mulligan
patrick.mulligan@iona.com

A Great Review

"Getting Focus()ed – and a Quick JavaScript Lesson," by Charles Arehart (Vol. 7, issue 1) was a good review of basic and useful concepts.

Satihs
bbabu@baan.com

The Benefits of DCG

"Dynamic Code Generation" by Norman Richards (Vol. 7, issue 2) is a great overall intro to the benefits and simple techniques of dynamic code generation.

Brian Maso
brian@blumenfeld-maso.com

Not a "Reel" Boss

"How's the Boss" by Bill Baloglu and Billy Palmieri (Vol. 7, issue 2) was a letdown after such a good topic selection. I've come across numerous different and eccentric managers with their own sets of problems. The classification given by the authors on the types of managers seems more like a story of a *reel* life rather than *real* life.

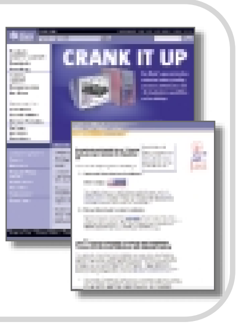
Ashish
aashish_agarwal@mailcity.com

Bill Baloglu and Billy Palmieri's column was a fine read – a great blend of psychology, staffing experience, wit, and wisdom.

Larry Mundo
Elle Mundo@aol.com

SUN CELEBRATES 20 YEARS OF INNOVATION

(Santa Clara, CA) – Sun Microsystems turned 20 this past March, celebrating two decades as an innovator whose focus on developing the products and technologies that make the Net work helped transform the Internet from an obscure sideshow into the ubiquitous tool it is today. For the past 20 years, Sun has been transforming the way computing occurs through new technologies. Today, as in the past, Sun is leveraging its philosophy of openness to foster customer choice to build their business success.
http://sun.com



InetSoft Releases Version 4.3 of Style Report

(Piscataway, NJ) – InetSoft Technology Corporation has released version 4.3 of Style Report. New features include a Web-based report designer, a full suite of debug/development tools, and reusable report components, allowing for easier report creation, easier deployment, and lower maintenance.
www.inetsoft.com

ReportMill5 Brings PDF and Flash to Java Web Apps

(Dallas) – ReportMill Software, Inc., has announced the general availability of ReportMill 5, a developer tool providing dynamic PDF and Flash Web pages and reports for Web applications. The new version is written entirely in Java and runs on all major platforms with support for all Java-based Web application servers, including those from BEA, IBM, Oracle, Sun, and Apple.
http://reportmill.com/webstart

ICS Releases Version 3.40 of Kronos Enterprise Scheduler

(Paramus, NJ) – Indus Consultancy Services (ICS) has announced the latest in a series of improvements to Kronos Enterprise Scheduler, a full-featured job-scheduling system written for the J2EE environment.

The new release fixes some minor bugs, and adds a new security feature to control user access to jobs, tasks, and schedules. A new environment setting allows the Kronos Enterprise Scheduler administrator to restrict users to seeing only those items they've created. Administrators still have full access to all items.
www.indcon.com

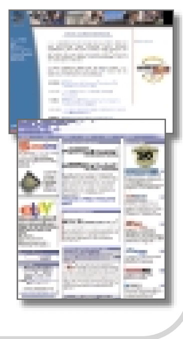
Slangsoft Licenses iTID Platform to SavaJe

(Boston) – SavaJe is integrating Slangsoft's intelligent text input and display platform with the SavaJe XE operating system. Device manufacturers (OEMs) can rely on the iTID-enabled SavaJe XE to create information appliances that meet today's demand for portable computing with fast, intuitive text input and scalable text display for up to 52 languages.
www.savaje.com

Rational Announces Support for BEA WebLogic Server 6.1

(Lexington, MA) – Rational Software has announced support for BEA WebLogic Server 6.1 with Rational XDE Professional: Java Platform Edition and a new BEA WebLogic Plug-In for the Rational Unified Process, simplifying and accelerating the development and deployment of Java and J2EE applications.

The BEA WebLogic Plug-In is available as a free download at the RUP Exchange on the Rational Developer Network (www.rational.net).
www.rational.com



eXcelon Announces Javlin 1.2

(Burlington, MA) – eXcelon Corporation has announced Javlin 1.2, a new version of its middle-tier J2EE data cache manager. The latest release features new functionality that enables tighter integration with application server environments, including BEA WebLogic Server. In addition, Javlin is supported on all WebLogic Server platforms, including Windows, Solaris, Linux, and HP-UX. Javlin 1.2 also now supports the JDK 1.3.
www.exceloncorp.com

Versata/ILOG Collaborate on Business Logic Initiative

(Oakland, CA) – Versata Inc. has announced an agreement with ILOG, calling for the companies to collaborate to offer the industry's first end-to-end business rules solution.

Under the terms of the agreement, Versata will develop a connector between the Versata Logic Server and ILOG JRules and, in turn, ILOG will develop a connector between their ILOG JViews for Workflow product and the Versata Logic Server.

This blend of business logic will enable customers to build highly transactional, performance-critical systems that integrate decisions or deduction rules that can be changed dynamically by business users.
www.ilog.com
www.versata.com

TOWER Technology Announces eProcess Objects Enhancements

(Boston) – TOWER Technology's TOWER eProcess Objects now include J2EE EJB middleware software.

With TOWER eProcess Objects, browser-based local and remote workers can participate in production-level workflows and integrated case management processes with no loss of functionality or performance and without the client maintenance and infrastructure costs of a private wide-area network.
www.towertech.com

EPOS PARTNERS WITH MONGOOSE TECHNOLOGY

(Auburn, AL / Houston) – EPOS Corporation and Mongoose Technology have teamed to add a portal offering as part of EPOS' FirstLine Encore platform of communications solutions. The EPOS portal product will be built upon Mongoose PortalStudio Enterprise Edition. In addition to enabling secure single-login access to all EPOS applications, the portal employs PortalStudio Search for accessing structured and unstructured content and provides a high degree of rule-based personalization using the Mongoose WebReactor.
www.mongoosetech.com
www.epos.com

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebSphere.

SYS-CON MEDIA

WebSphereDevelopersJournal.com

WebSphere DEVELOPER'S JOURNAL

Introductory Charter Subscription

SUBSCRIBE NOW AND SAVE $31.00 OFF THE ANNUAL NEWSSTAND RATE

ONLY $149 FOR 1 YEAR (12 ISSUES) REGULAR RATE $180

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Do You Have Access to the Internet?

The World's Leading Independent WebSphere Developer Resource

Then Subscribe Online and Save $31!

It's that easy.

EnginData Presents

2002 Developer Market Survey Reports

$2,995
Java Developer Market Survey 2002
$2,495
Web Services/XML Developer Market Survey 2002
$1,995
Wireless Developer Market Survey 2002

Our comprehensive reports offer insight and strategy to guide your most critical business decisions in today's fastest growing technologies...

✓ Establish your product and marketing strategy

✓ Understand your customer's needs

✓ Evaluate Technology & Trends

engindata.com

engindata✓ RESEARCH

JAVA DEVELOPER'S JOURNAL

The World's Leading Java Resource

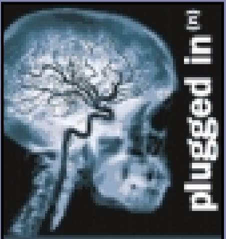
HOME | ADVERTISE | SUBSCRIBE | CONTACT | SYS-CON MEDIA   www.javadevelopersjournal.com

DOT.COM
- buyer's guide
- java forums
- mailing list
- java jobs
- java store

MAGAZINE
- advertise
- authors
- customer service
- editorial board
- subscribe

CONTENT
- archives
- digital edition
- editorial
- features
- interviews
- product reviews
- source code

J2ME | J2SE | J2EE | Home | Online



THE JAVA DEVELOPER CONNECTION™ PROGRAM

Get pre-release Java™ technology downloads

REGISTER TO GET THEM

Java Sun

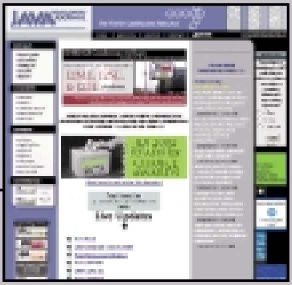
What's Online... April 2002

JavaDevelopersJournal.com

Visit www.javadevelopersjournal.com and learn about the latest news and events from the world's leading Java resource. Know what's happening in the industry, minute by minute, and stay ahead of the competition.

2002 Readers' Choice Awards

Make your opinion count. Vote today for the ***Java Developer's Journal*** **Readers' Choice Awards**, which have earned the reputation as the most respected industry awards of its kind. Known as the "Oscars of the Software Industry," this year's awards feature an expanded list of product categories and other additions that will make the fifth annual Readers' Choice Awards the best so far!



Web Services Edge 2002 Conference and Expo

Sign up for the Web Services Edge 2002 East International Conference and Expo, June 24–27, at the Jacob Javits Convention Center in New York City. This year's event is expected to be the biggest and best ever with the top IT professionals in the business speaking about the hottest industry topics. The Web Services Edge 2002 Conference and Expo will feature presentations prepared for a diverse audience. Whether you consider yourself a beginner or at the top of the IT world, this conference is for you!

Developing SOAP Web Services

It takes only one day to become a Web services developer. Sign up today for the Web Services Edge 2002 World Tour Series in New York City on April 19, or San Francisco on May 13, where you can network with the experts and learn to develop and deploy Web services. This one-day event will cover base technology to more advanced issues. Every attendee receives a *free* copy of all software used.

Java Discussion Groups

What are the hottest topics in the Java discussion groups this week? You can read the latest discussions from the biggest IT professionals in the industry, search archived messages, or post your own comments at www.sys-con.com/java. No matter what your preference, the Java discussion groups are a great way to stay on top of the most important issues in the IT world!

Free Seminar and Software

Now ***JDJ*** fans have the opportunity to learn more about Oracle9*i* Application Server by visiting www.javadevelopersjournal.com. Oracle9*i* Application Server includes over 250 new features to make your J2EE applications more scalable, reliable, and secure. You can sign up to attend an Internet seminar featuring Oracle9*i* Application Server's chief architect and download a preview copy of their software for free.

WAKESOFT Learn more now!

Click for a FREE 30-day trial AltoWeb

M-1 Mobile App Server aligo Download Now!

sitraka

Get SonicMQ® sonic SOFTWARE

MERANT DataDirect

The Truth About Web Services

KADA

spiritsoft going beyond jms

InstallAnywhere PowerUpdate ZERO G www.ZEROG.COM Click here to try

SHOP ONLINE AT **JDJSTORE.COM** FOR BEST PRICES OR CALL YOUR ORDER IN AT **1-888-303-JAVA**

BUY THOUSANDS OF PRODUCTS AT GUARANTEED LOWEST PRICES!

GUARANTEED BEST PRICES
FOR ALL YOUR
WEB SERVICES
SOFTWARE NEEDS

JDJStore.com

MICROSOFT

$2,238.99 **Visual Studio .NET Enterprise Architect**

Visual Studio .NET provides developers with the most productive tool for building next-generation applications for Microsoft Windows® and the Web. Visual Studio .NET Enterprise Architect (VSEA) builds on the power of Visual Studio .NET Enterprise Developer by including additional capabilities for designing, specifying, and communicating application architecture and functionality. It enables software architects and senior developers to provide architectural guidance and share best practices across the development team.

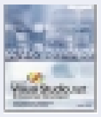
SYS-CON MEDIA

$79.00 **Web Services Resource CD**

The Most Complete Library of Exclusive *WSJ* and *XML-J* Articles on one CD! This CD is edited by well-known *WSJ* Editor-in-Chief Sean Rhody and organized into more than 40 chapters containing more than 400 exclusive *WSJ* & *XML-J* articles.
$100 Coupon Inside for Web Services Edge Conference & Expo!

MICROSOFT

$1,629.99 **Visual Studio .NET Enterprise Developer**



With Visual Studio .NET Enterprise Developer, developers can securely version and share their source code, share best practices, target scalable .NET Enterprise Servers, choose from a wide range of third-party tools and technologies, and easily tune the performance of their Web applications and Web Services through the extensive performance-testing tools in Visual Studio .NET.

WHIZLABS

$39.95 **WebSphere@Whiz Certification Simulator**

3 Mock Tests (159 Questions). It comes with a test engine and a question bank of 159 questions on the latest pattern of the IBM WebSphere Certification Exam. It also consists of a diagnostic test which will help you know your strengths and weaknesses so you can plan your preparation accordingly.

ALTOWEB

$3,540.00 **Application Platform Release 2.5**

The AltoWeb Application Platform lets you build, deploy, and manage J2EE applications and Web services up to 10x faster without requiring extensive J2EE or Web services expertise. How? By replacing lengthy, custom and complex J2EE, XML and Web services coding with rapid component assembly and reuse.

SILVERSTREAM

$495.00 **Extend Application Server Developer Edition (5 User)**

SilverStream eXtend is the first comprehensive, real-world development environment for creating Web Services and J2EE applications. The seamless integration of our proven eBusiness engines and designers gives you the benefits of XML-based, enterprise-wide integration and the power to create, assemble and deploy service-oriented applications.

W W W . J D J S T O R E . C O M   OFFERS SUBJECT TO CHANGE WITHOUT NOTICE

Once you're in it...

- Wireless Business & Technology
- Java Developer's Journal
- XML-Journal
- ColdFusion Developer's Journal
- PowerBuilder Developer's Journal

reprint it...

Contact Carrie Gebert
201 802-3026
carrieg@sys-con.com

SYS-CON MEDIA

RePrints

Your Own Magazine

Do you need to differentiate yourself from your competitors?
Do you need to get closer to your customers and top prospects?
Could your customer database stand a bit of improvement?
Could your company brand and product brands benefit from a higher profile?
Would you like to work more closely with your third-party marketing partners?
Or, would you simply like to be a magazine publisher?

SYS-CON Custom Media is a new division of SYS-CON, the world's leading publisher of Internet technology Web sites, print magazines, and journals.

SYS-CON was named America's fastest-growing, privately held publishing company by *Inc. 500* in 1999.

SYS-CON Custom Media can produce inserts, supplements, or full-scale turnkey print magazines for your company. Nothing beats your own print magazine for sheer impact on your customers' desks... and a print publication can also drive new prospects and business to your Web site.

Talk to us!

We work closely with your marketing department to produce targeted, top-notch editorial and design. We can handle your distribution and database requirements, take care of all production demands, and work with your marketing partners to develop advertising revenue that can subsidize your magazine.

So contact us today!

East of the Rockies,
Robyn Forma,
robyn@sys-con.com,
Tel: 201-802-3022

West of the Rockies,
Roger Strukhoff,
roger@sys-con.com,
Tel: 925-244-9109

SYS-CON CUSTOM MEDIA

The Next Big Thing

Getting up to speed

WRITTEN BY
BILL BALOGLU & BILLY PALMIERI

In the simplest of terms, the technologies behind Web services (SOAP, UDDI, and XML) combine to connect computers to each other and to perform complex tasks without human interaction.

Web services applications are expected to be broad and far-reaching. Early uses include companies utilizing them within their business to integrate multiple applications and systems; to find a common way to link payroll, sales, and CRM applications; and to communicate with each other.

B2B applications will enable business and trading partners to integrate their systems through a shared registry (think intranet) or public registry (think Internet) of information.

Consumer applications will enable computers to search Web sites for information (such as travel dates and fares) as well as open up communications with the desired source and perform transactions. An early model of the same concept was Napster's peer-to-peer connecting of one computer to another to share and exchange information.

The new piece is the actual transaction that will be possible through Web services. Skeptics might predict that this will enable your computer to not only find the wrong book online, but to purchase and have it delivered to your doorstep in record time.

While consumer applications of Web services may be a few steps down the road, companies are understandably excited about the technology's potential to integrate critical functions both in- and outside their organizations.

Simple Object Access Protocol (SOAP) and Xquery (which queries databases using XML) tap into Universal Description Discovery and Integration (UDDI), a giant repository of Web services. The twofold function of UDDI is to be a global registry of all public Web services and also the technical standard that defines an interface to the registry.

HP, IBM, Microsoft, Sun, and SAP are all running an instance of the global UDDI registry and share the database that replicates on a daily basis. Microsoft's XML Web services platform is called .NET, and Sun has dubbed their version Sun ONE (Open Net Environment), an "open framework Web service using the Java architecture."

An early explorer of Web services technologies is Tony Hong, who (with his brother James Hong) cofounded the Web services site, www.xmethods.net, in August 2000.

"The site started as a developer-focused grass roots project," says Hong, who was director of engineering at a B2B software solutions company called Ventro (now NexPrise) in Mountain View, California.

"I was responsible for internal and external integration projects and helping companies integrate transactions between their trading partners. The Web services stack was built to integrate systems and I saw great potential in the technology," says Hong.

After leaving Ventro, Hong looked into other ways of experimenting with Web services technologies although the concept of the UDDI registry hadn't yet come about. He and his brother James bootstrapped the xmethods.net site as a repository of Web services information that enables developers to experiment.

"On the site, Web services applications are contributed to by the developer community," says Hong. "Some are demos, some are commercial, but all of them are working."

From our perspective on the staffing side of the industry, we're hearing a lot about the demand for Web services technologies from both our senior engineers and hiring manager clients. Hong agrees that this is where a large portion of the high-tech industry is headed.

How does an ambitious engineer get up to speed on the *Next Big Thing*?

There are a number of books about Web services and SOAP. There's a great book list at www.soaplite.com and a comprehensive view of the industry at www.webservices.org.

From a skills development perspective, at the nuts-and-bolts platform level you should learn XML and how to deal with basic Internet protocols like HTTP. Then you'll have the foundation to tackle the Web services protocols. You should also know the mechanics of Internet access.

"Once the plumbing of Web services is in place, people will wonder what to do with it, and B2B integration will be one of the first uses," says Hong. "The B2B integration space will see a lot of new activity, so an understanding of B2B integration and process mapping will be important."

Who will be hiring engineers with strong Web services skills?

At first, the platform and tools builders like BEA, IBM, and Sun should have a strong need for people with these skills. Eventually, enterprise application vendors like Oracle, SAP, and PeopleSoft will also need engineers with Web services skills.

Online sources of information about Web services include www.webservices.org, www.uddi.org, www.xmethods.net, and www.microsoft.com/net.

jdjcolumn@objectfocus.com

AUTHOR BIOS

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

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

ADVERTISERINDEX

ADVERTISER	URL	PHONE	PAGE
Actuate Corporation	www.actuate.com/info/jbjad.asp	800-884-8665	57
Altova	www.altova.com		31
AltoWeb	www.altoweb.com		33
Apptricity	www.apptricity.com	214-596-0601	89
Aquarius Solutions	www.aquariussolutions.com		72
Ashnasoft Corporation	www.ashnasoft.com		85
BEA	www.bea.com/download		4
Borland Software Corp.	www.borland.com/new/optimizeit/94000.html		71
Canoo Engineering AG	www.canoo.com/ulc/	41 61 228 94 44	19
Capella University	www.capellauniversity.edu	1-888-CAPELLA	43
Compoze Software	www.compoze.com/jdj	866-266-7693	87
Compuware Corp.	www.compuware.com/products/optimalj	800-468-6342	25
DataDirect Technologies	www.datadirect-technologies.com	800-876-3101	11
Dice.com	www.dice.com		49
Dynamic Buyer Incorporated	www.ibm.com/smallbusiness/dynamicbuyer		97
EnginData Research	www.engindata.com		109
eXcelon Corporation	www.exln.com	800-962-9620	29
Fiorano Software	www.fiorano.com/tifosi/freedownload.htm	800-663-3621	79
IBM	ibm.com/websphere/ibmtools		34
IBM	ibm.com/websphere/ibmtools		35
IBM	ibm.com/db2/outperform		37
ILOG	www.ilog.com/jdj/jrules	1-877-223-ILOG	55
InetSoft Technology Corp.	www.inetsoft.com/jdj	888-216-2353	73
Infragistics, Inc.	www.infragistics.com	800-231-8588	14-15, 83
InstallShield Software Corp.	www.installshield.com	847-240-9111	53
INT, Inc	www.int.com	713-975-7434	18
Interland	interland.com	1-800-308-6145	41
Java Developer's Journal	www.sys-con.com	800-513-7111	88, 99
LOOX Software Inc.	www.loox.com	800-684-LOOX	61
Metrowerks Corp.	www.wireless-studio.com		9
Mongoose Technology	www.portalstudio.com		39
New Atlanta Communications	www.newatlanta.com		59
Northwoods Software Corporation	www.nwoods.com	800-226-4662	72
Oracle Corporation	oracle.com/javacode	800-633-1072	13
ParaSoft Corporation	www.parasoft.com/jdj4	888-305-0041	51
PointBase, Inc.	www.jdj5.pointbase.com	877-238-8798	65
Pramati Technologies	www.pramati.com	877-PRAMATI	75
Precise Software	www.precise.com/jdj	800-310-4777	27
QUALCOMM Incorporated	www.brew2002.com		63
Quintessence Systems Limited	www.in2j.com		67
Rational Software	www.rational.com/offer/javacd2		47
SilverStream Software	www.silverstream.com/challenge		21
Sitraka	www.sitraka.com/jprobe/jdj	800-663-4723	17
Sitraka	www.sitraka.com/jclass/jdj	800-663-4723	81
Sitraka	www.sitraka.com/performasure/jdj	800-663-4723	116
Simplex Knowledge Company	skc.com	845-620-3700	95
Softwired, Inc.	www.softwired-inc.com	41-14452370	77
Sonic Software	www.sonicsoftware.com	800-989-3773	2
SpiritSoft	www.spiritsoft.com/climber		23
SOLUTIONSsite	www.solutionssite.com	877-838-9500 x122	16
SYS-CON Industry Newsletters	www.sys-con.com	201-802-3020	98
SYS-CON Subscription Offer	www.sys-con.com/suboffer.cfm		107
Thought Inc.	www.thoughtinc.com	415-836-9199	69
TogetherSoft Corporation	www.togethersoft.com/1/jd.jsp	919-833-5550	6
WebGain, Inc.	www.webgain.com/toplink_create3.html	1-877-WebGain Ext.15858	115
WebLogic Developer's Journal	www.sys-con.com/weblogic	800-513-7111	107
Web Services Edge World Tour 2002	www.sys-con.com	201-802-3069	104-105
Web Services Journal	www.sys-con.com	800-513-7111	96, 101
WebSphere Developer's Journal	webspheredevelopersjournal.com	800-513-7111	109
Wireless Business & Technology	www.sys-con.com	800-513-7111	94
XML-Journal Subscribe	www.sys-con.com	800-513-7111	90
Zero G	www.zerog.com	415-512-7771	3

Next Month

CAST JavaMiner

JavaMiner may be the answer for people who need to communicate a working application when time is limited.

reviewed by Jason Bell

Programming Neural Networks in Java

This article shows a simple, yet particle, neural network that can recognize handwritten letters, and describes the implementation of a neural network in a small sample program.

by Jeff Heaton

Ask Dr. Java

The answers to your Java-related questions.

by James McGovern

Manifest Destiny

This article presents some of the issues involved with packaging Java code. Specifically, it explores the Java manifest file and suggests ways that it can be used to manage JAR file dependencies and eliminate classpath issues normally associated with cross-platform deployment.

by Norman Richards

Using the Java Native Interface Productively

Although we try to make our applications pure Java, outside forces sometimes make this impossible. This article discusses supporting a C/C++ API in Java to enable a Java application to use it.

by Andrew J. Chalk

Forces of the Universe and Other Sundries

WRITTEN BY
BLAIR WYMAN

I've been thinking that, if I want to keep writing these monthly bits o' fluff, I'd better start making some sense pretty soon. If you've been reading Cubist Threads, you know I'm prone to launching into some banal diatribe about the prosaic minutiae of my midwestern upbringing.

I'm just not sure that these recollections are exactly what your long-suffering editor had in mind when he gave me this fun gig, so perhaps I'll save the flowery stories and try to be a bit more practical.

Oh, bother...the very notion of making sense is fundamentally at odds with my manifest propensity for waxing preposterous: endeavoring to elevate the perfect absence of meaningful content to its epitome. It has been a joyful exercise in the art of putting the "blank" back in "blank verse."

Lately, I'm thinking I owe you something more substantial than that, and frankly it bothers me sometimes. I'm afraid I am squandering my tenuous opportunity to push a political agenda, or to advocate a belief system, or to fire your imagination and tell you how white your shirts can be.

On the other hand, you probably already have a perfectly serviceable belief system, a solid set of conscientious political convictions, and dazzlingly incandescent unmentionables. Maybe I should just chill out on the whole heavy, head-shaping trip and simply hope my words will make you smile, or laugh, or maybe even dream a little.

I like to think I can keep an open mind – perhaps it's the rust gathering thickly on the door hinges – but I'm surely kidding myself. It's much more likely that my belief system is quietly calcifying into a spiny little ball in some tangential tidepool, viciously insulating itself against anything that dares to tread nearby.

But sometimes, thankfully, something still comes along to shake my convictions to their roots. For instance, several years back now, the whole concept of object-oriented programming was one such mindbender. Before my first exposure to OOP, I was perfectly happy in my procedural understanding of computer programming. I didn't want to hear about any object-oriented approach to programming – an approach so new that it must surely be deeply and hopelessly flawed. After all, what could possibly be more concise and accurate than a well-drawn flowchart?

Then I read a skinny little book about OOP – one of the multivolume set of documentation that came with a later version of Turbo Pascal – and some glaring lights started illuminating my previously comfortable ignorance. OOP made a lot of sense, right away, but was too abstract for me to grasp at first reading. OOP didn't really "click" for me until later, when I needed a general-purpose numeric input field in a GUI I was writing for plotting some interesting chaos simulator or somesuch. I thought about creating a numeric input field "object" – an object I could put anywhere on the screen, and from which I could get input – and OOP fit my requirement perfectly. I was sold on the potential of OOP and haven't looked back.

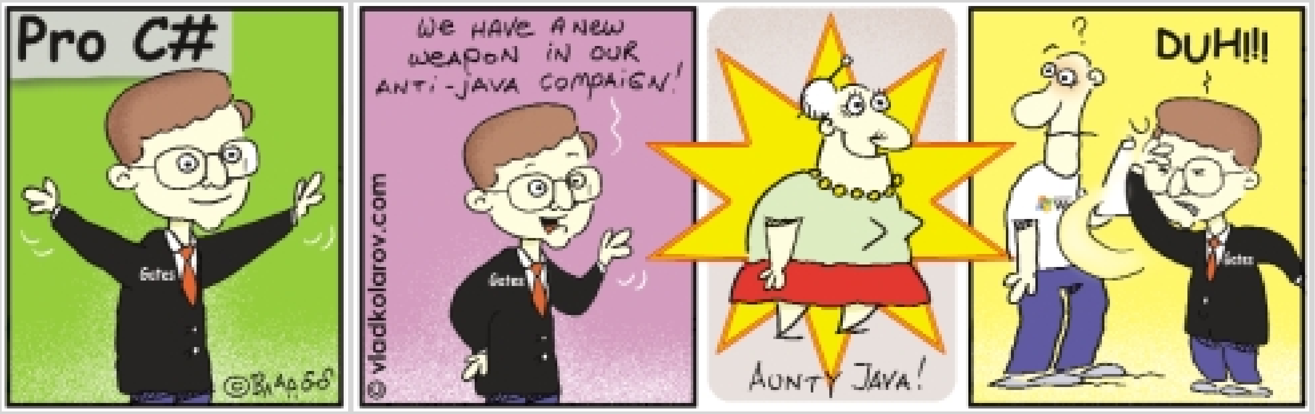
I've recently had another such pseudoawakening: a couple of weeks ago a colleague of mine recommended a book called *The Elegant Universe* by Brian Greene. The book purports to be a layman's look at string theory, the 11-dimensional description of reality that holds some promise for logically unifying all the forces in the universe, and my skepticism was unmaskable. I've read a bit of Einstein's work on relativity, beginning pretentiously and finishing prematurely, and sort of vaguely "getting it." I have trouble imagining the macrouniverse in any other terms.

While I haven't finished reading the book yet, I soon will; I'm excited at the prospect of my spiny little ignorance urchin being upended and eviscerated yet again. With luck, I will have lubricated my brain doors a little and shooed away some ignorance in the process. But fear not, gentle reader, that I may run out of ignorance; while I hope to chip away at it for years to come, my personal supply knows no bounds.

blair@blairwyman.com

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

Java Dudes



WebGain, Inc.

www.webgain.com/toplink_create3.html

Sitraka

www.sitraka.com/performasure/jdj