

BEA WebLogic

DEVELOPER'S JOURNAL

Volume:1 Issue:5

weblogicdevelopersjournal.com

web services EDGE world tour 2002

SAN FRANCISCO, CAMAY 13

Page 46 Register for Web Services Edge 2002 East Gold Passport, Attend the World Tour FREE!

FROM THE EDITOR Migration Strategies by Jason Westra page 5

GUEST EDITORIAL Tracking Transactions by Ethan Henry page 6

Full Conference Program INSIDE ▽ page 27

THE FUTURE OF ENTERPRISE COMPUTING

SAVE \$200

Conference: June 24-27 Expo: June 25-27

Call Today!

JACOB JAVITS

RETAILERS PLEASE DISPLAY UNTIL JUNE 30, 2002

\$15.00US \$16.00CAN

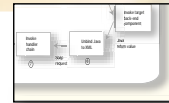


SYS-CON MEDIA



Viresh Garg 12

TRANSACTION MANAGEMENT: Transactions and Beans Why have a dog and bark yourself?



8

Peter Holditch

SAM'S SOAPBOX: Migrate Early, and Often Maintain your flexibility to upgrade when the time comes



20

Sam Pullara

WLDJ FEATURE: Out with the Old, In With the New Porting to WebLogic Server 7.0



22

Nick Tran

ARCHITECTURE: The WebLogic Workshop Architecture A new concept in a familiar style



32

Tyler Jewell

DEADLOCKS: Common WLS Deadlocks and How to Avoid Them Avoid common application mistakes - and let your threads play out



36

Rob Woollen

CERTIFICATION: Getting Started Becoming a BEA Certified developer on WebLogic Server 6.0



42

Dave Cooke

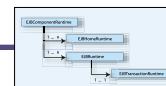
TIPS AND TRICKS: The Benefits of SQLj Replacing JDBC code with SQL - making life easier



46

Mika Rinne

WLDJ FEATURE: Exploring WebLogic JMX PART 2 Creating WLS management applications



50

Dan Mackinnon

BEA

www.developer.bea.com

BEA

www.developer.bea.com

Wily Technology

www.wilytech.com

EDITORIAL ADVISORY BOARD
TYLER JEWELL, FLOYD MARINESCU,
SEAN RHODY

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JASON WESTRA

EDITORIAL DIRECTOR
JEREMY GEELAN

EXECUTIVE EDITOR
GAIL SCHULTZ

MANAGING EDITOR
CHERYL VAN SISE

SENIOR EDITOR
M'LOU PINKHAM

EDITOR
NANCY VALENTINE

ASSOCIATE EDITOR
JAMIE MATUSOW

ASSOCIATE EDITOR
JEAN CASSIDY

WRITERS IN THIS ISSUE

DAVE COOKE, VIRESH GARG, ETHAN HENRY,
PETER HOLDITCH, TYLER JEWELL, MIKA RINNE,
DAN MACKINNON, SAM PULLARA, NICK TRAN,
JASON WESTRA, ROB WOOLEN,

SUBSCRIPTIONS

For subscriptions and requests for Bulk Orders, please send your letters to Subscription Department.
SUBSCRIPTION HOTLINE:
SUBSCRIBE@SYS-CON.COM
Cover Price: \$15/Issue
Domestic: \$149/YR (12 Issues)
Canada/Mexico: \$169/YR
Overseas: \$179/YR
(U.S. Banks or Money Orders)

PUBLISHER, PRESIDENT AND CEO
FUAT A. KIRCAALI

VP, PRODUCTION & DESIGN
JIM MORGAN

SENIOR VP, SALES & MARKETING
CARMEN GONZALEZ

VP, SALES & MARKETING
MILES SILVERMAN

VP, EVENTS
CATHY WALTERS

VP, BUSINESS DEVELOPMENT
GRISHA DAVIDA

CHIEF FINANCIAL OFFICER
BRUCE KANNER

ASSISTANT CONTROLLER
JUDITH CALNAN

ACCOUNTS PAYABLE
JOAN LAROSE

ACCOUNTS RECEIVABLE
JAN BRAIDECH

ACCOUNTING CLERK
BETTY WHITE

ADVERTISING ACCOUNT MANAGERS
MEGAN RING • ROBYN FORMA

ASSOCIATE SALES MANAGERS
CARRIE GEBERT • ALISA CATALANO
KRISTIN KUHNLE • LEAH HITTMAN

CONFERENCE MANAGER
MICHAEL LYNCH

EXECUTIVES, EXHIBITS
MICHAEL PESICK • RICHARD ANDERSON

ART DIRECTOR
ALEX BOTERO

ASSOCIATE ART DIRECTORS
AARATHI VENKATARAMAN • LOUIS CUFFARI
CATHRYN BURAK • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR
TAMMI BEATTY

WEBMASTER
ROBERT DIAMOND

WEB DESIGNERS
STEPHEN KILMURRAY • CHRISTOPHER CROCE,

CONTENT EDITOR
LIN GOETZ

JDSTORE.COM
ANTHONY D. SPITZER

CUSTOMER SERVICE
ANTHONY D. SPITZER

EDITORIAL OFFICES

SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 592-3900 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
BEA WebLogic Developer's Journal (ISSN# 1535-9581)
is published monthly (12 times a year)
Postmaster: Send Address Changes to
BEA WEBLOGIC DEVELOPER'S JOURNAL,
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645



BY JASON WESTRA
EDITOR-IN-CHIEF

Migration Strategies for WebLogic Server

“Migration,” in terms of J2EE, generally means good

things for a project or application. It means bug fixes from a previous version, new features to make your life easier (whether you are a developer or a system administrator), and often it means performance, fault-tolerance, and scalability enhancements.

So, if migration means good things, why does the term “migration” usually send chills down the spines of all of us – CIOs, project managers, and developers alike? In our minds, migration equates to missed project deadlines, and product features and application functionality that may have to be put on hold as highly valuable resources such as senior developers, system administrators, and QAs are busy ensuring the migration goes smoothly.

There are two types of users in this world performing “migrations” with J2EE products. First, there are those that built their application on WebLogic Server and WebLogic Portal/Commerce Server and are currently migrating toward the new BEA WebLogic Platform 7.0. These users are moving to the new version to take advantage of a new feature that will lower development time dramatically (enough to make the time spent on migration worthwhile). I recall migrating to WebLogic 6.1 to get Message-Driven Bean (MDBs) support. Our team was sick of writing JMS (Java Messaging Service) code in startup classes and writing this “plumbing” was time-consuming. For our application, MDBs were a huge time-saver and well worth the upgrade.

Second, there are the poor souls who mistakenly built their application on another J2EE server and are migrating from it to WebLogic for its superior scalability, performance, and ease of use. These users have applications that most likely took advantage of proprietary extensions that need to be torn out and replaced with a WebLogic-specific

ic, or standard J2EE equivalent. As you'll see in this issue, even an application that contains no proprietary extension code can be a bear to migrate to another J2EE-compliant server. Studying migration strategies can not only help you plan for your next migration, it can help you be a better architect and developer because you learn the touch points that J2EE applications have between different servers. You learn how sound design and coding practices can make upgrades and migration between servers faster and painless.

This month in *WLDJ* we focus on migration strategies for WebLogic Server that covers both types of J2EE migrations. This issue contains articles to help you think through your migration plan when moving your application to the latest version of WebLogic, as well as how to move from another application server to WebLogic Server. Viresh Garg from BEA contributes to this month's theme by guiding us through the process of moving from WLS 6.x to WLS 7.0. Along with full J2EE 1.3 support, the new version of our favorite server offers tight integration of the portal and commerce products under a complete e-business platform and security enhancements touted as 1.5 years ahead of the competition. Cool stuff!

Also, this month we have Nick Tran's article on migrating Sun's J2EE Blueprints to WLS. Sun's J2EE Blueprints are generally deployed on the Sun J2EE RI (reference implementation), but we'd never want to deploy a production application on anything but WebLogic, right? So, let's get that application originally built on another server migrated over to WebLogic and deployed on a production-ready server.

I encourage those of you currently planning to migrate to the new BEA WebLogic Platform or just beginning new development on the latest version of WebLogic, to take a look inside. This month is full of helpful advice and sound practices for everyone developing on WebLogic.



AUTHOR BIO...

Jason Westra is the editor-in-chief of *WLDJ* and the CTO of Evolution Hosting, a J2EE Web-hosting firm. Jason has vast experience with the BEA WebLogic Server Platform and was a columnist for *Java Developers Journal* for two years, where he shared his WebLogic experiences with readers.

CONTACT:jason@sys-con.com

© COPYRIGHT 2002 BY SYS-CON PUBLICATIONS, INC. ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM, WITHOUT WRITTEN PERMISSION. FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR. SYS-CON PUBLICATIONS, INC. RESERVES THE RIGHT TO REUSE, RE-PUBLISH AND AUTHORIZE THE READERS TO USE THE ARTICLES SUBMITTED FOR PUBLICATION. ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES. SYS-CON PUBLICATIONS, INC., IS NOT AFFILIATED WITH THE COMPANIES OR PRODUCTS COVERED IN WEBLOGIC DEVELOPER'S JOURNAL.

BY ETHAN HENRY



Tracking the Transaction: Performance-Centric J2EE Development

To build a J2EE system that's both extensible and maintainable it's important to take a functional approach. By separating the system into tiers, you ensure that your Web tier is built to handle the presentation layer, the EJB layer can manage business logic, and the database is built to store data persistently and take care of most of the complex data concurrency issues. While this functional model is accepted practice, the problem is that it neglects what makes an e-business system successful: fast transactions.

The reality is that well-designed J2EE systems aren't necessarily high-performance systems. This reality usually hits home in production as operations teams struggle to meet service-level commitments on transaction times. The costs can soar alarmingly quickly. One response may be to add to your application server cluster, or invest in more hardware. But a proactive approach to performance in preproduction can mitigate risk and protect your IT group from these later costs. You'll need to balance the functional approach with a performance-centric approach, supported by appropriate tools.

Adopting a performance-centric approach involves analyzing transaction execution paths through the entire system, and tuning your system to accelerate throughput and response time. Focus your tuning on the transactions that are genuinely taxing system resources and failing to meet response time requirements. With a transaction-centric focus, you also run less

risk of optimizing areas that aren't negatively impacting your customer experience.

One important way to improve performance is via caching. Intelligent use of caching is arguably the single most important factor in delivering fast transaction processing. But to implement caching effectively you must understand the transaction path. Otherwise, you may implement caching on a process that is already inherently fast or that is infrequently used – a poor use of development resources.


To improve transactional efficiency, reduce the time spent in expensive data computation and lookup by caching data forward in the presentation or business logic layer. Completed results of business logic can be kept forward in the presentation layer to save on the cost of recomputing the same piece of data for multiple requests. Session handling should also be kept as far forward as possible in the presentation layer. This ensures that the business logic can be implemented using high-performance stateless logic (typically, stateless session EJBs). Use caching to lower costs, especially on your most expensive infrastructure investments: databases and clustered application server machines.

Let's revisit the limitations of accepted practice. In large-scale systems, you may incur significant performance degradation if you take the golden principles of object-oriented programming too literally. Fine-grained and tightly coupled design patterns can be

communication-heavy, slowing down transaction response time. Try experimenting with newer, performance-centric patterns such as a more loosely coupled, coarse-grained design for component interactions. Remember to focus on the transaction path so you can target enhancements where you'll reap a worthwhile performance gain.

Finally, let's talk tools. Many tool vendors are making noise about transaction response times. But a closer look reveals that the analysis some tools provide is in fact highly tier-specific, leaving you with only a partial view of transaction throughput. Production-monitoring tools, for example, provide response-time metrics without any data to identify where in the transaction execution path the performance is degrading.

Applying these monitors to preproduction analysis work is going to leave you with more questions than answers about your transaction processing. So build your diagnostic arsenal carefully, with tools that can give you both broad visibility through your assembled system as well as deep performance intelligence within the J2EE application itself.

With a more flexible approach to J2EE design patterns, the right tools, and a healthy obsession with the transaction execution path, you'll be well on your way to delivering a high-performance, not just well-designed, system. Operations folks will thank you for it. And your customers will be getting what they need, when they need it. 



AUTHOR BIO...

Ethan Henry is manager of Training Services and a former product manager at Sitraka. He has been involved with Java technology since 1995 as a Java instructor, a Java developer, and a "Java Evangelist." Ethan has written a number of articles on Java technology and has spoken at numerous conferences, including JavaOne.

CONTACT: ethan.henry@sitraka.com

Precise

www.precise.com/wldj



That was what an old girlfriend periodically said to me. Needless to say, we're no longer together – I wasn't keen on the comparison. "Shall I compare thee to a dog?" is rather less poetic than I like. But in thinking about this month's transaction column, the comment seemed strangely germane to the world of app servers and transactions.

Transactions and Beans

WHY HAVE A DOG AND BARK YOURSELF?

BY PETER HOLDITCH



AUTHOR BIO...

Peter Holditch joined BEA as a consultant in the Northern European Professional Services organization in September 1996. He now works as a presales architect in the UK. Peter has a degree in electronic and computer engineering from the University of Birmingham.

CONTACT...

peter.holditch@bea.com

I've mentioned several times in these articles that the underlying purpose of an application server is to provide out-of-the-box functionality that you would otherwise have to write and maintain yourself, despite it having only indirect relevance to the business problem at hand. For example, "Transfer money from my current account to my savings account" is an operation that any banker would comprehend. "Implement a two-phase commit transaction manager" would probably leave them dazed and confused, despite the implicit requirement for an atomic transfer of funds.

Bankers never seem keen on creating money (unless it's created in their pockets) or making it vanish (unless it vanishes from your pocket and reappears in theirs). With this in mind, the wise developer buys an app server and uses its JTA transaction manager to provide the necessary atomicity guarantees underlying the business operation, while the unwise (and shortly to be unemployed) one goes off on a two-year mission to implement transaction services, because it's fun to do.

I'll Be Your Dog

So, it's clear that we don't want to waste our lives (and risk our jobs) by writing a transaction manager. Wouldn't it be great if we could go one step fur-

ther and not write a single line of code about transactions and have them magically handled by the infrastructure? Not only would that mean fewer lines of code to understand and maintain, but it would guarantee that all our code made a consistent set of assumptions about the transactional semantics of the application. This would, in turn, make reusing the application componentry much easier without having to go back to the design docs (or worse, the code) and decipher what the underlying intentions and assumptions were.

Happily, they thought of all that when J2EE was invented, and the EJB container is equipped not only to manage persistence and security, but transactions too. In fact, not only is it equipped to manage transactions, but for entity beans it's mandatory that the container manages the transactional behavior – bean-managed transactions are only an option for session or message-driven beans (MDBs), but I'm getting ahead of myself.

What Are the Options?

In general in the EJB world, transactions can be container or bean managed. As you would expect, for container-managed transactions (CMTs), all the code to control the starting, committing, and aborting of transactions is provided and managed by the EJB container. For bean-managed transactions (BMTs), the bean implementer gets to use the JTA API set to control the transaction demarcation directly. As I mentioned, entity beans aren't permitted to use bean-managed transactions (presumably to encourage reuse of entity beans). In fact, the EJB specification states that any transaction in force when a BMT method is called should be suspended by the EJB container before the business logic is invoked. Thus, using BMT beans in various transactional contexts isn't possible.

Bean reuse is really maximized by using container-managed transactions. Additionally, when using message-driven beans, unless you use container-managed transactions, there can be no transactional integrity between the dequeuing of the JMS message the bean is about to process and any updates that the bean's `onMessage` method may cause to happen. You get the general picture.... Aside from those comments, I'll mention BMTs no further, since this article is about getting the application server to do the transactional work for you.

In the Beginning There Was the Transaction – Container Managed...

There are six different policies that can be associated with methods on a CMT bean: `NotSupported`,

Sitraka

www.sitraka.com/jclass/wldj



Required, Supports, RequiresNew, Mandatory, and Never. Code executed in methods marked NotSupported and Never will not execute in the context of an active transaction. If such a method is called in the context of a transaction, the transaction will be suspended in the NotSupported case and a RemoteException will be thrown to the caller in the Never case. Either way, any work done by the method will not cause more work to be added to a transaction.

Required and Mandatory methods, on the other hand, work in the context of an existing transaction. For the Required case, the container will start a new transaction before calling the method if no transaction is associated with the calling context. For the Mandatory case, if the method is called outside the context of a transaction, a TransactionRequiredException will be thrown. Methods marked RequiresNew will have a fresh transaction started for them by the container. Any caller's transaction will be suspended first. In either event, the method will always execute in the context of a transaction. Finally, the Supports policy allows the code to participate in any existing transaction context or in none, if it's called outside of a transaction scope.

As a general rule of thumb, I would consider it good practice to avoid the use of policies that leave you uncertain as to a method's transactional context. In particular, avoid Supports. In the same way that uninitialized variables can cause hard-to-find bugs in systems (especially when code is reused) it's much better to state, "This must be used in a transaction" or "This can't be" rather than "Well, I'm not really sure."

So What Happened in the End?

So, it's all very well. You've told the container when to start transactions for you, but it can't see your business logic.... How does it know when a transaction should end, and whether it should end with a commit or an abort? In the general case, if the container started a transaction, "T," before invoking a method, "M," then it will commit "T" before returning the results of the Method to the requestor.

There are two exceptions to this behavior. The first occurs when the method throws a system exception. The second, an application exception, "should be used to communicate application-level problems to the client, such as "You can't withdraw that much money, it would put you in overdraft."

There's no reason for the container to assume you want to roll back a transaction because of an application exception. It's a common misconception that throwing any exception from a method will cause the current transaction to be rolled back. It won't. A System exception is an exception not covered by the application exception category. These typically indicate catastrophic technical failures – "I couldn't get a database connection", for

example. If one of these is thrown, the transaction will be rolled back (or marked for rollback, if the container didn't start it). Technically, the container considers any exception that is a subclass of RuntimeException or RemoteException to be a system exception.

So, what if you want the current transaction to roll back, but you want to throw an application exception? Well, there's a method on the EJBContext interface called setRollbackOnly(). This method will mark a transaction rollback only, and if the transaction was started by the container, then calling this method from your business logic indicates that the transaction should be rolled back rather than committed when the method returns.


While we're on the subject, there's another EJBContext method that pertains to transactions: getRollbackOnly(). Clearly, if your bean has been called in the context of a transaction that's already doomed, then there's no point wasting lots of CPU cycles doing some complex calculation with results that will only be rolled back and thrown away. If your beans contain such CPU-intensive code, then you can introduce a good optimization only by checking if the transaction is marked rollback before you execute it.

Got the Message?!

So that's transactions and beans. Now, one final word about message-driven beans. For CMT message-driven beans, the dequeuing of the JMS message that gets passed to your bean's onMessage method is done in the same transaction context that your method is invoked in. If your code causes the transaction to be rolled back, by any of the means discussed above, then (thanks to the transaction manager and its good old ACID properties), it will be as if the message never got dequeued.

Of course, if the message hasn't been dequeued then it's still on the queue waiting to be consumed so the very next thing that will happen is that it will be dequeued and redelivered to your waiting onMessage method, which will throw an exception, the transaction will be rolled back... ad infinitum.

So be careful if you're writing MDBs – better to write errant messages to some dead-letter queue for your application and commit the transaction than throw an exception and abort it. Unless, that is, you get some kind of warped pleasure from watching app servers spinning around infinite loops. WebLogic's JMS implementation also provides some configurable parameters to handle this case, to delay redelivery of a rolled-back message, or to automatically place it on a dead-letter queue after some number of retries.

That's all for now. The nice men in white coats are coming to commit me but remember – if you're using an app server it should be barking, not you.... See you next month. 

"How does it know when a transaction should end, and whether it should end with a commit or an abort?"

Mongoose

www.mongooseotech.com



BY
VIRESH GARG

Upgrading to WebLogic

YOU DON'T HAVE TO REWRITE EXISTING

AUTHOR BIO...

Viresh Garg is a senior software engineer with BEA Systems. He has seven years' experience in software development with distributed systems using object-oriented technologies. Viresh has been working for BEA systems in the WebLogic Server division for nearly three years in various support and engineering roles.

CONTACT...

vgarg@bea.com

The much awaited WebLogic Server 7.0 beta release was announced on Feb. 24, 2002, at BEA's eWorld conference in San Diego. Continuing in its path as the Number 1 Web application server, WLS 7.0 implements J2EE 1.3 technologies, Web services, and other Internet standards to provide a reliable framework for highly available, scalable, and secure applications.

WebLogic Server's seamless integration of disparate, heterogeneous platforms and applications enables your network to leverage existing software investments and share the enterprise-class services and data crucial to building mission-critical e-business applications. WLS 7.0 comes with very powerful features and enhancements in the area of Web services, clustered JMS, distributed destinations, and administration. BEA has completely rewritten the security infrastructure, J2EE connectors, and JCA adapters, and is offering some interesting new development tools.



Server 7.0

APPLICATIONS

Not only does it make sense to choose WLS 7.0 as the platform for enterprise e-business and integration application development and deployment, but also to upgrade from prior versions of WLS, whether 4.5.1, 5.1, or 6.1.

Special emphasis was placed on making sure that upgrading or migration to 7.0 is easy, straightforward, cost-effective, and, most important, that it doesn't require any code change in existing applications. WebLogic Server 7.0 comes bundled with power utilities that help with migration and upgrading and minimize the programmer

or administrator's required intervention. This article makes an attempt to walk you through the step-by-step process to upgrade existing applications to WLS 7.0.

Upgrade from WLS 6.0/6.1 to WLS 7.0

The good news for upgrades from these versions is that you'll need to make little or no change to startup scripts and configuration repositories. Depending on installation, classpath options may need little or no change. Its no longer necessary to include the license file in the classpath. You

should also have startup scripts in the domain directory itself. For the JVM version, WLS 7.0 installs 1.3.1_02 with server installation.

INSTALLATION/CONFIGURATION CHANGES

To provide greater flexibility in terms of configuration and management of application code and server systems, WebLogic Server 7.0 comes with a new structure for the domain directory. You're no longer required to have domains in the WebLogic Server installation directory. In fact, it's recommended to move the domains to a different directory while upgrading from 6.1 to 7.0. This directory structure change also comes with new startup scripts. Look for example startup scripts in the sample domains (PetStore and examples) created as part of installation.

Moving directory structure comes with ease of maintenance later on in the application life cycle but can be tricky while upgrading. Make sure that while moving directories, you identify all references to filepaths and resolve them to new filepaths. This is not an automated process.

APPLICATION/DEPLOYMENT CHANGES

For the most part, all applications, exploded or archived, that were deployed in 6.x versions should work just fine with WLS 7.0. However, some considerations should be taken into account when upgrading applications.

Consider using local interfaces when migrating an EJB application from 6.0; they were included in 6.1. Also remote relations are not supported in 7.0. To upgrade an application from 6.0 to 7.0, you have to bundle all EJBs in EJB 2.0 CMP relations into one JAR file.

If WebLogic Server is used as a proxy to another WebLogic Server or cluster, it's recommended that you change the names of proxy servlets from `weblogic.servlet.internal.HttpClusterServlet` to `weblogic.servlet.proxy.HttpClusterServlet`, and from `weblogic.t3.svr:-HttpProxyServlet` to `weblogic.servlet.proxy.HttpProxyServlet`.

The WLS 7.0 distribution no longer includes the unmodified Xerces parser and Xalan transformer. However, WebLogic Server 7.0 is bundled with WebLogic FastParser, specially written for SOAP and WSDL (small-to-medium) XML documents. Consider changing Xalan API references to JAXP. This will require code change but will help in future upgrades and maintenance of applications, as you won't be tied to vendor-specific code.

WebLogic's Java-COM bridge, JCOM, has also been upgraded. Its migration is very simple, as you're no longer required to

FIGURE 1

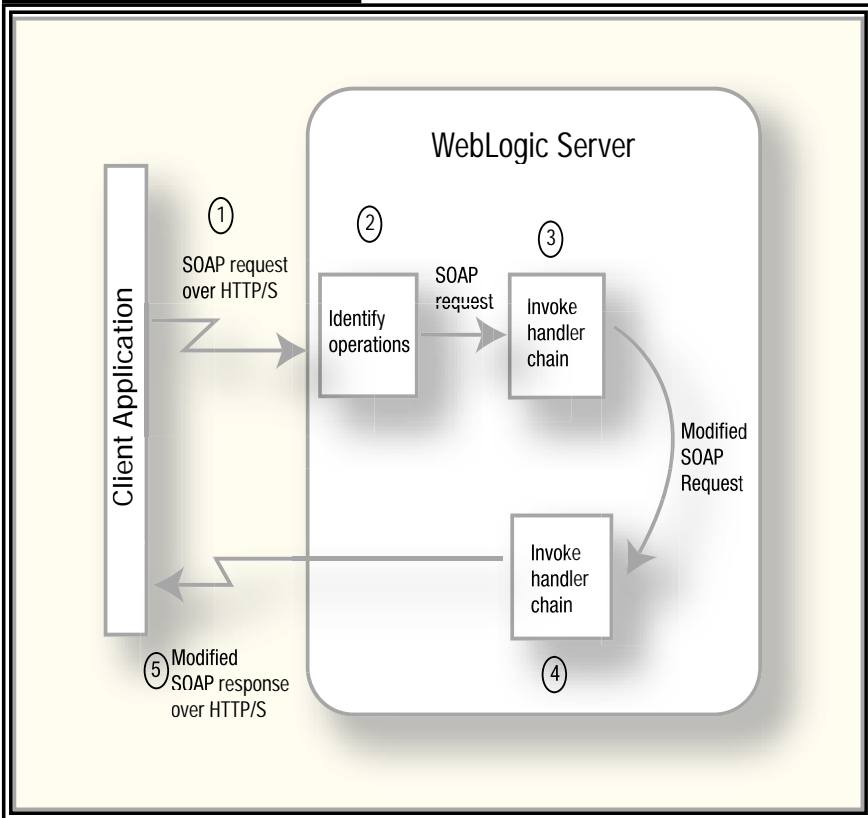
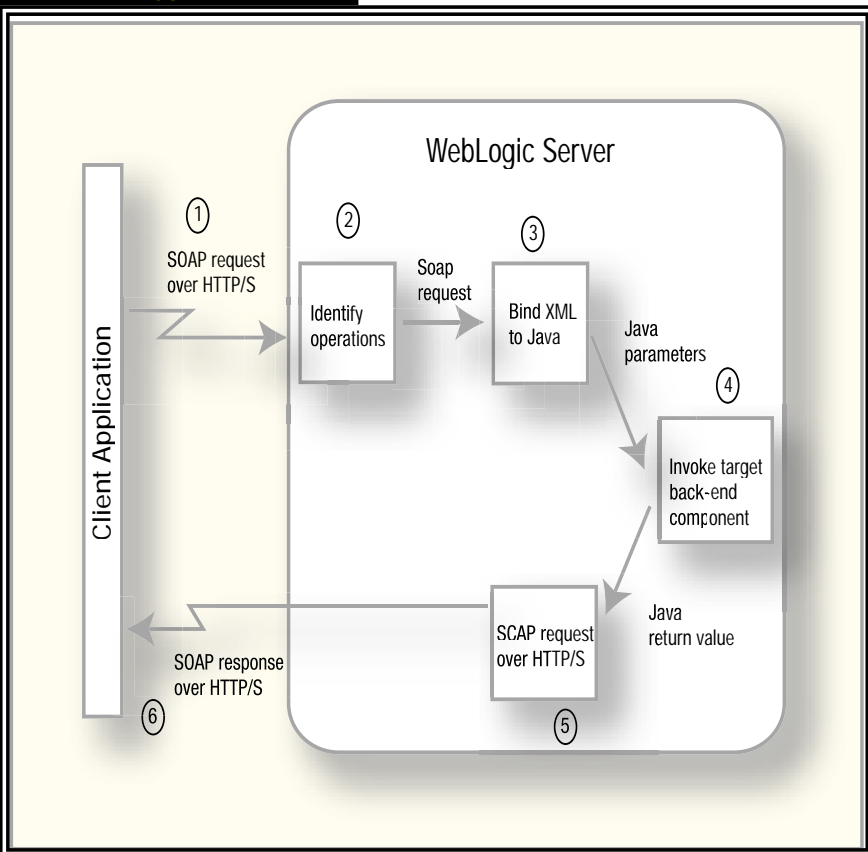


FIGURE 2



write/install a bridge. Most of the command-line properties are no longer required or are configurable from the console.

Upgrading the security information is probably the single most complex upgrade in the exercise. WebLogic Server 7.0 ships with an embedded LDAP server enabled on the admin server in the domain. WLS 7.0 can automatically run in “compatibility” mode, allowing you to keep earlier versions’ file-based security configuration for users, groups, and ACLs. It’s up to customers to decide whether to keep the 6.1-based security realms or to upgrade to the 7.0 security framework to take advantage of some of the new features, such as a fully compliant JAAS authentication model and better configuration and administration for security constraints for applications. GA release of WLS 7.0 will provide automatic migration utilities to upgrade earlier version file-based security information to the new WebLogic Security Realm stored in an embedded LDAP server. However any storage medium can be used for authentication and authorization.

An application using “NULL” or “guest” user must change to specify a valid username. Alternatively for a guest user, you can configure “guest” as a valid user; however, a NULL user is no longer supported. This might require a code change in an application, but in general, production applications use valid security credentials and this change won’t be required.

Web services have been improved to use the new JAX RPC-based API. The new model for client-side programming is standards-based and assumes that a Web services client can be written using the same API and standards using JAX RPC specification. WLS 7.0 even provides an optional Java client JAR file that includes all the classes, interfaces, and stubs that are needed on the client side. This JAR file includes the generic implementation of the JAX-RPC specification as well as Web service-specific implementations to minimize the amount of Java code needed to invoke a particular Web service. Benefits aside, however, this improvement means that your Web services clients rewritten on 6.1-based Web services will have to be rewritten based on new JAX RPC-based APIs.

A WebLogic Web service has to be assembled and packaged in an enterprise archive EAR file. This includes assembling and packaging all the components of the service (such as the EJB JAR file, the handler classes, etc.) and generating the web-services.xml deployment descriptor file in a single deployable EAR file. Changes and enhancements in Web services runtime require repackaging for EAR files. WebLogic 6.1 provided the wsgen Ant task to automatically assemble all components for a Web Service in an EAR file. WebLogic 7.0 provides the servicegen Ant task to assemble an RPC-based Web service. For upgrading an RPC style 6.1 Web-Service component rep-

Sitraka

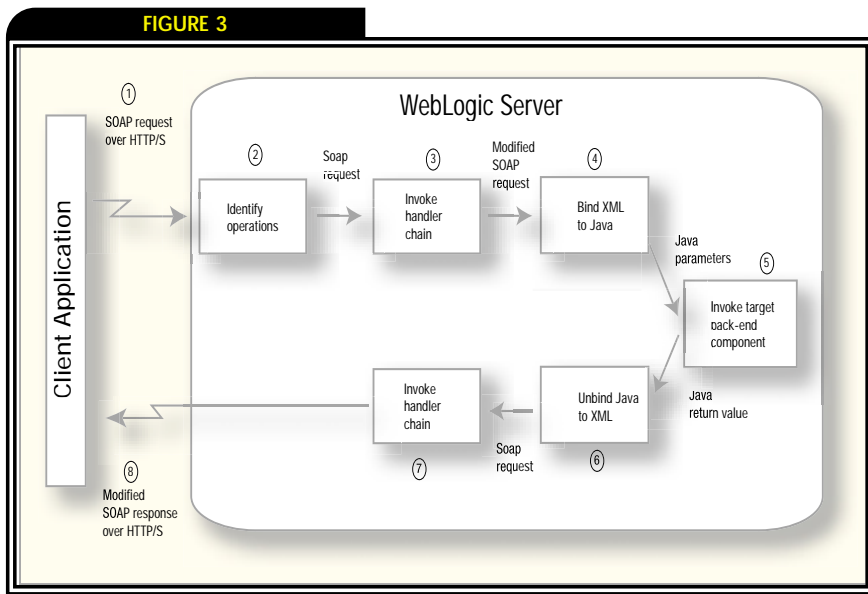
www.sitraka.com/jprobe/wldj

represented by an EAR file, the EAR should be reassembled using the new build XML script and Ant task.

Due to support for multiple back ends, the use of message-style Web services is no longer recommended. As the new Ant task, servicegen, and the new build script only support RPC-style Web services, message-style Web services can be upgraded in one of two ways. Either they can be rewritten using new multiple back-end support or a new EAR can be generated manually by carefully extracting the components from the old EAR file and editing deployment descriptors.

WebLogic Server 7.0 supports two-phase application deployment to ensure homogenous application deployment in clusters and also for better status reporting for success/failure of deployments. Old-style deployment will continue to work "as is" in WebLogic 7.0; however, use of the two-phase deployment model is recommended, as it is flexible and rich in terms of new features, such as application ordering, application-scoped configuration (a JDBC resource, for example), better deployment tracking and status, improved redeployment without loss of service, and flexible multiple options for application staging.

- While upgrading or designing new JMS applications, consider distributed destinations. By enabling you to configure multiple physical destinations as members of a single distributed destination set, WebLogic JMS offers a level of redundancy, and therefore service continuity, in the event of a single Weblogic Server failure within a WebLogic Server cluster. Once properly configured, your producers and consumers are able to send and receive to the distributed destination. WebLogic JMS then distributes the messaging load across all available destination members within the distributed destination. When a destination member becomes unavailable, traffic is then redirected toward other available destination members in the set. However, producers and consumers that are pinned to a failed destination member must be closed and re-created.
- Upgrade your servlet code to take advantage of the new Servlet 2.3 spec ServletRequest/ResponseWrapper objects. I strongly recommend using these objects in your design as some of the old code may not compile if it's dependent on undocumented internal WebLogic implementation details.
- While upgrading Web services, consider changing existing Web services applications to allow asynchronous (one-way) requests, access to request/response SOAP messages for further processing by handlers in the chain, user defined data types and SOAP attachment handling, all available with the new WLS 7.0 architecture. Figures 1–3 depict the various architectures supported by WebLogic Server 7.0.
- Selectively expose EJB methods via Web services rather than exposing all methods (a system limitation in 6.1 required you to expose all methods in a remote interface of an EJB).
- Consider configuring the staging and active directory name for each managed server. Consider using different staging modes that affect the file distribution for applications and allow using storage data networks or shared file systems for application staging. These are new parameters in support of the new two-phase deployment model. Old-style behavior can be achieved using the two-phase attribute on the ApplicationMBean defining the application.



The use-case model

SUGGESTED DESIGN CHANGES TO IMPLEMENT DURING UPGRADE

- WebLogic JMS takes advantage of the migration framework implemented in the WebLogic Server core for clustered environments. This allows WebLogic JMS to properly respond to migration requests and bring a JMS server online and offline in an orderly fashion. This includes both scheduled migrations as well as migrations in response to a WebLogic Server failure. When upgrading, restructure your JMS framework to take advantage of these features.

Upgrading from WLS 4.5.1/5.1 to WLS 7.0

INSTALLATION/CONFIGURATION CHANGES

You must upgrade the existing license file from 4.5.1/5.1 to 6.1. This can be done online by submitting the old XML license file to <http://web.support.beasys.com/custsupp>. You can then run the utility UpdateLicense from the WebLogic bin directory to merge the newly obtained license with the existing license file.

WebLogic 4.5.1 and 5.1 used weblogic.properties files (global, per cluster, and per server

weblogic.properties files) as configuration repositories for each individual server. From WLS 6.0 onward all configuration information for all servers in an administrative domain is stored in a single configuration repository named `config.xml`, which resides in the domain root directory on the admin server machine of the domain. The first task in upgrading is to convert all `weblogic.properties` files to a single `config.xml` file. This process seems very complex, tedious, and error-prone, but WebLogic provides conversion utilities and help from a GUI-based console that make it very easy and intuitive.

Start WebLogic Server 7.0 and follow the “convert `weblogic.properties`” hyperlink, which will navigate you through the entire process to convert all `weblogic.properties` files to a single domain `config.xml` file. During conversion all security information from `weblogic.properties` files is stored in the 6.1 style file `Realm.properties`. Servlets, JSPs, and other classes in the document roots of 4.5.1 and 5.1 are assembled in a Web application by a conversion utility and the `web.xml` and `weblogic.xml` files for the new Web application are generated.

Due to major overhauls in classloading in WLS 6.0 and above, startup scripts must change, as there is no more WebLogic classloader, and `weblogic.class.path` isn't required as a system property. Export of transitive closures of interfaces and class references in interfaces to WebLogic classloader made it difficult for EJBs to evolve in 4.5.1/5.1. The new classloading model provides more flexibility in terms of what can change in a running server. The new model supports a single classloader for each application, thereby allowing a single application as a deploy-ment/redeployment unit.

WLS 6.0 and above provides new configuration and administration models, thereby requiring some rework around writing new startup scripts for admin and managed servers. There is little in terms of migration that can be used from existing 4.5.1 and 5.1 startup scripts.

APPLICATION/DEPLOYMENT CHANGES

- Assemble the Web component in the format shown below to deploy a Web application as an individual component or part of a J2EE application archive.

```
WebApplicationRoot\(\Publicly available files such as
| .jsp, .html, .jpg, .gif)
|
|           +WEB-INF\--+
|
+ classes\(\directory con
|         taining Java classes
|         including servlets
|         used by the
|         Web Application)
|
+ lib\(\directory containing
```

```
|           JAR files used by the
|           Web Application)
|
+ web.xml
|
+ weblogic.xml
```

- Session cookie information and session persistence properties have moved from `weblogic.properties` to `weblogic.xml` of the respective Web application. This allows multiple Web applications to have different session attributes. The conversion utility automatically creates a `weblogic.xml` for the Web application generated by conversion utility.
- The WLS 7.0 EJB container supports EJB 1.1 and EJB 2.0 beans. However it is recommended that new development be done with EJB2.0. WebLogic 5.1 supported EJB 1.1, so EJBs deployed in WebLogic 5.1 can be directly deployed in WLS 7.0, which has a stricter XML parser. Some errors allowed by earlier versions are no longer permitted.
- You can upgrade EJB 1.0 deployment descrip-

“ Not only does it make sense to choose WLS 7.0 as the platform for enterprise e-business and integration application development and deployment, but also to upgrade from prior versions of WLS, whether 4.5.1, 5.1, or 6.1 ”

tors used in WLS 4.5.1 to EJB 2.0 using the `DDCreator` and `DDConverter` utilities, but first those descriptors must be upgraded to 1.1. Using `DDConverter` utilities, WLS 5.1 deployment descriptors can be upgraded to 7.0 to take advantage of new features in WLS 7.0.

- An EJB deployment includes a standard deployment descriptor in the `ejb-jar.xml` file. The `ejb-jar.xml` must conform to either the EJB 1.1 DTD (document type definition) or the EJB 2.0 DTD.
- An EJB deployment requires the `weblogic-ejb-jar.xml` file, a WLS-specific deployment descriptor that includes configuration information for the EJB container. This file must conform to the WLS 5.1 DTD or the WLS 7.0 DTD. In order to specify the mappings to the database, container-managed persistence entity beans require a CMP deployment descriptor that conforms to the WLS 5.1 CMP DTD, the WLS 7.0 EJB 1.1 DTD, or the WLS 7.0 EJB 2.0 DTD.

- For a JMS upgrade follow these steps:
 1. WebLogic 7.0 provides a JMS configuration conversion utility that should be used to port the 4.5.1/5.1 style weblogic.properties-based JMS configuration to the new config.xml-based configuration. Once your configuration information is converted, the JMS administrator needs to review the resulting configuration to ensure that the conversion meets the needs of the application, and adjust the values, if necessary.
 2. In WLS 5.1, the JMS data and durable subscriber information was kept in five database tables that were accessible via JDBC. In WLS 7.0, JMS queues are defined during configuration, and no longer saved within database tables. Message data and durable subscriptions are stored either in two JDBC tables or in a directory within the file system. The JDBC database store content format from WLS 5.1 is not compatible with WLS 7.0.
 3. Update existing code, as required, to reflect the feature functionality changes. For example, the createQueue() and createTopic() methods do not create destinations dynamically; they create only references to destinations that already exist, given the vendor-specific destination name.
 - 4. Start up WebLogic Server, and the existing JDBC database stores are ported automatically. If the automatic porting fails for any reason, it is retried the next time WebLogic Server boots.
 - Use java.rmi.Remote instead of weblogic.rmi.Remote and java.rmi.*Exception instead of weblogic.rmi.*Exception. Also, weblogic.rmic must be used to generate new stubs and skeletons. Old stubs and skeletons are not compatible with WLS 7.0.
 - For security, the conversion utility creates the FileRealm.properties file from user, group, and ACL information in the weblogic.properties file(s). This gets you to the point of 6.1-style security realm. To convert this file to a WebLogic Server 7.0-style WebLogic Realm and to store this information in embedded LDAP or another security repository, follow the instructions for upgrade from 6.x to 7.0.
- A number of APIs were deprecated in WLS 7.0, so consider changing your design to use a recommended replacement API or deleting the API and rewriting your code to accommodate the change (see Table 1).

TABLE 1

DEALING WITH DEPRECATED APIS

These APIs are deprecated in WLS 7.0, so consider a design change to remove it or instead use the recommended API:

Deprecated API	Recommendation
Administrative console GUI	Remove and change code
Deployer Tool	Remove and change code
Jview support	Remove and change code
RemoteT3	Remove and change code
SSI	Remove and change code
T3Client	Remove and change code
WebLogic Beans	Remove and change code
WebLogic Bean Bar	Remove and change code
WebLogic COM	Remove and change code
weblogic.deploy	Weblogic.Deployer
WebLogic Enterprise Connectivity	Use JCA Adapters
WebLogic Events	Use JMS
WebLogic HTMLKona	Use JSP
WebLogic JDBC t3 Driver	Use RMI driver and DataSources
WebLogic jHTML	Remove and change code
WebLogic Remote	Remove and change code
WebLogic Server Tour	Remove and change code
WebLogic Time Services	weblogic.management.timer
WorkSpaces	JNDI or Stateful RMI/Stateful session bean/HTTP Session
Zero Administration Client	Use JavaWebStart.

Deprecated APIs

RECOMMENDED DESIGN CHANGES DURING UPGRADE

It is recommended that you introduce a new server instance that will act as administrator of the entire domain. Segregate the administration and application work to different server instances.

You may wish to use J2EE-compliant application assembly and deployment rules rather than individual component deployment. WLS 7.0, being a J2EE-compliant application server, supports application deployment for enterprise archives or individual components such as Web applications, EJB, and J2EE connectors. These applications and components can be deployed in exploded or archived format (.ear, .war, .rar, or .jar depending on component). A format for an enterprise application is shown below.

```
EnterpriseApplicationStagingDirectory\
|
+ .jar files
|
+ .war files
|
+ .rar files
|
+META-INF\-*
|
+ application.xml
```

You may wish to use local interfaces to boost performance and bundle EJB and Web applications in a single application archive while upgrading.

Consider better abstraction for database entities using powerful object-to-relational mapping provided by the EJB 2.0 CMP relational model. The CMP relational model is easier to configure and manage in WLS 7.0 using WebLogic Builder, and comes with huge performance enhancements for relational database queries.

Consider using Data sources and TxDataSources for JDBC access from EJB instead of the JDBC 1.0 API or the WebLogic connection pool using pool or JTA JDBC drivers provided by WebLogic. Data sources allow your applications to be more configurable, as well as portable, across server platforms.

You may wish to reconfigure your JMS applications to take advantage of new attributes, templates, and logical abstractions in JMS applications. JMS is a clustered service from version 6.0 onward. Also review the design considerations in the JMS upgrade from 6.x to 7.0 (all of them apply here too).

Using user-defined queues for dispatching incoming requests may be appropriate. User-defined queues were introduced as a publicly supported feature in 6.1.

“ Special emphasis was placed on making sure that upgrading or migration to 7.0 is easy, straightforward, cost-effective and, most important, that it doesn't require any code change in existing applications ”

Conclusion

All existing applications, with the exception of Web services, will continue to work fine on WLS 7.0, without any penalty on performance or features that were available in earlier versions. However, as described in this article, you should consider the proposed design changes to optimize new architecture and features. Migration is a good opportunity to consider making such changes. There is a small learning curve in understanding the new security and deployment model and distributed JMS destinations.

I think some time and resources spent on re-architecting applications heavily based on those features will pay off in the form of flexibility, standards-based implementation, and overall better reliability, scalability, and availability for the system. As always, BEA Professional Services and BEA Support can help with any specific migration questions and/or migration assignments. ●

SAVE \$31 Off*

the annual newsstand rate




ANNUAL NEWSSTAND RATE
\$190.00
YOU PAY
\$149.00
YOU SAVE
\$31 Off the Annual Newsstand Rate

Receive 12 issues of *PBDJ* for only \$149.00. That's a savings of \$31 off the annual newsstand rate. Visit our site at www.powerbuilderjournal.com or call 1-800-513-7111 and subscribe today!

Here's what you'll find in every issue of *PBDJ*

- New PowerBuilder Features
- Tips, Tricks and Techniques in Server-Side Programming
- DB Programming Techniques
- Tips on Creating Live PowerBuilder Sites
- Product Reviews
- Display Ads of the Best Add-on Products
- PowerBuilder User Group Info

*Offer subject to change without notice



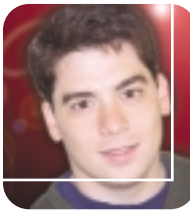
Migrate Early, and Often

MAINTAIN YOUR FLEXIBILITY TO UPGRADE
WHEN THE TIME COMES

BY SAM PULLARA

WORKING AT A SOFTWARE COMPANY AND watching products evolve over the years, “Migrate Early, and Often” is the best advice I can give someone who is trying to stay on the cutting edge of technology. Obviously, if you have an application that’s in production with no problems and you aren’t planning on adding any new features, then of course keep the same version of the underlying platform. However, if you’re developing a new application and are currently in the process of architecting and designing, you should definitely talk with your vendor about what you can expect in upcoming releases. Although you might not target the initial phase of your project to the new release, you can ensure that you have the flexibility to upgrade when that time comes. Keeping your options open will, in the long run, give you a more stable and maintainable product. As one of my friends, Peter Seibel (now at Kenamea), said, “Software gets better.” And I think that you can say that almost universally. Some might argue that it should be phrased, “Software gets bigger,” but I take exception to that. This might be true in the world of desktop applications, where new releases inevitably exist only to encourage people to buy the next upgrade; in the enterprise people have support contracts that ensure the next version is available. All the critical aspects of enterprise software increase with each new release: reliability, administration, scalability, and performance. Those are the normal ones, but I’m going to add one more aspect that also gets better with age – one that WebLogic has concentrated on in our newest release: usability. That may not be part of the big four, but it’s fast becoming one of the more important measures in the enterprise software world.

So what has WebLogic done in this release to increase usability? Migration is an excellent example and the focus of this issue of *WLDJ*. One of the big problems that people had moving from previous WebLogic Server versions to the newest release was the changes that had to be made to accommodate J2EE. The move from EJB 1.0 to EJB 1.1 was particularly painful because there were some source changes that had to be made. For people who are migrating from 5.1, the biggest obstacle was often the switch from using the `weblogic.properties` file to using Web applications (.war files). This was especially true for our ISVs, who had become accustomed to releasing their products in a certain way. In that transition we attempted to solve the problem



through documentation and support. Although people got through it, I’m glad that future migrations will be done mostly through effective applications of technology. From WebLogic Server 6.0 and forward we have kept the `config.xml` file as a backwards-compatible configuration store so new versions of the server can read an old version of the configuration and still understand it. In the process it upgrades that configuration to be perfectly compatible with the new system. WebLogic Server 7.0 helps you migrate to the newest version using a number of tools and built-in functionality.

Inside the server, we again have the capability to read an old `config.xml` and upgrade it to the newest version. In fact, you might have heard about the new security framework that was released with WebLogic 7.0. This framework can even read your old security properties from 6.1. This gives everyone an easy upgrade path if you’re not planning to use many of the features of the new security framework right away. To address the needs of applications, WebLogic Builder will automatically upgrade your old application `.ears`, `EJB .jars`, and Web Application `.wars` to the newest versions of the specifications. You do this by starting up the tool, loading the application in question, adding in any new functionality, validating the configuration with the (possibly) new restrictions, and then saving the configuration files back to the application. This upgrade path for applications makes it much easier for people to test on the new platform to ensure that everything behaves as it did before the upgrade.

Even with these tools, however, the developer still has some responsibility to ensure that the application works properly with the new system. Some changes to the J2EE specification, plus bug fixes that occur between releases, can sometimes cause incompatibilities that must be found prior to going to production with an application. In order to find these differences you must have a proper set of unit, feature, system, and performance tests in place to validate that the new version of the platform software performs as you would expect. If properly implemented, these tests can find any problems well before they are deployed to a customer.

Finally, there’s one other selfish reason to upgrade to the next release of the product. You’ll be working with the current state of the art. Staying on top of new technologies, like J2EE 1.3 or Web services, is very important for any developer. To sum up what I’ve said, I will rephrase the tagline: “Migrate as soon as you can.”



AUTHOR BIO...

Sam Pullara has been a software engineer at WebLogic since 1996 and has contributed to the architecture, design, and implementation of many aspects of the application server.

CONTACT: sam@sampullara.com

ReportMill

www.reportmill.com



BY
NICK TRAN

AUTHOR BIO...

Nick Tran is an applications engineer at BEA systems and the lead for the WLS samples team. He oversees the development of core API examples for WLS and porting of end-to-end applications to WLS. Before working at BEA, Nick worked as a software consultant building enterprise applications.

CONTACT...

nick.tran@bea.com



As one of the two Sun J2EE Blueprints applications, the Java Smart Ticket Demo application, like its well-known sibling, Pet Store, illustrates best practices for designing J2EE applications.

Port Store illustrates best practices for larger J2EE applications, and the Java Smart Ticket application illustrates best practices for designing wireless applications. Both Blueprints applications were designed with portability in mind. The Java Smart Ticket application is a less complicated example application for showing what it takes to port an application to WebLogic Server 7.0.

The Java Smart Ticket application runs without modification on the J2EE 1.3 Reference Implementation server. Since WebLogic Server 7.0 is fully compliant with the J2EE 1.3 specifications, getting the Java Smart Ticket application to run on WebLogic Server requires minimal effort.

The porting takes place in three stages:

1. Porting the EJB tier
2. Porting the Web Application (WebApp) tier
3. Putting it all together.

The last step, which is the most involved, includes building the Enterprise Application Archives (EAR) file, creating tables and uploading them to the PointBase server, and deploying and configuring the application for WebLogic Server 7.0. Very little Java code has to be changed in order to port the Smart Ticket

application to WebLogic Server 7.0. In fact, only

one Java class file must be changed, and this has more to do with the differences in the databases than with application servers. WebLogic Server 7.0 ships with PointBase 4.2, a pure Java RDBMS; the J2EE Reference Implementation server ships with Cloudscape.

The last section of this article will discuss running the application with the help of the phone emulator in the Java 2 Micro Edition (J2ME) Tool Kit.

Now let's start porting!

Getting Started

The instructions given here are for Windows. If you're using UNIX, please substitute / for \ in path names and % with \$ in variables appropriately.

Download and install WebLogic Server 7.0 from <http://commerce.bea.com/downloads/products.jsp>. Throughout the article, I will use %SAMPLES_HOME% to refer to the samples home directory, `bea_home\weblogic700\samples`.

Download and install the J2ME Tool Kit from <http://java.sun.com/products/j2mewtoolkit>.

Download the Java Smart Ticket Demo 1.1 from <http://developer.java.sun.com/developer/releases/smartticket>. Wherever you unzip `smartticket-1_1.zip`, we will refer to that as

%SMARTICKET_HOME%. I placed mine under the %SAMPLES_HOME%\server\src directory.

How to Port the Enterprise JavaBeans

Note: The `ejb-jar.xml` file that Sun ships uses an EJB 1.1.dtd, which does not support the `<unchecked>` element in the `<method-permission>` stanzas. The `<unchecked>` element can be found instead in the EJB 2.0 specification. Since the four `<method-permission>` stanzas don't restrict access to any of the methods on the beans, they can safely be commented out without affecting the application (see Listing 1; the code for this article can be found on our Web site at www.sys-con.com/weblogic/sourcecode.cfm).

The first task in porting any Enterprise JavaBeans (EJB) is to determine the type of EJB. This can be done by looking in the `ejb-jar.xml` for any bean. The Java Smart Ticket `ejb-jar.xml` file is located in %SMARTICKET_HOME%\src. The application employs three 1.1 stateless session beans and one Bean Managed Persistence (BMP) entity bean. These are relatively easy to port.

WebLogic-specific EJB settings are contained within the `weblogic-ejb-jar.xml` and `weblogic-cmp-rdbms-jar.xml` files. These files contain configuration information specific to WebLogic and are not included with the Java Smart Ticket application. In WebLogic Server 7.0, the `weblogic-ejb-jar.xml` file is no longer required by all EJBs. However, the `weblogic-cmp-rdbms-jar.xml` file is still required by all container managed persistent beans. We will need only the `weblogic-ejb-jar.xml` file.

Each of the EJBs should have a similar XML stanza contained in the `weblogic-ejb-jar.xml` file. The MovieInfo EJB, for example, has the following XML stanza in its `weblogic-ejb-jar.xml` file (see also Listing 2):

```
<weblogic-enterprise-bean>
  <ejb-name>MovieInfo</ejb-name>
  <reference-descriptor>
    <resource-description>
      <res-ref-name>jdbc/MovieInfoDataSource</res-ref-name>
      <jndi-name>MovieInfoDataSource</jndi-name>
    </resource-description>
  </reference-descriptor>
  <jndi-name>Smarticket.MovieInfoEJB</jndi-name>
</weblogic-enterprise-bean>
```

Here the `ejb-name` element, `MovieInfo`, corresponds to the `ejb-name` element with the same name in the `ejb-jar.xml` file. This is how you specify which of the EJBs defined in the `ejb-jar.xml` file you wish to extend in the `weblogic-ejb-jar.xml` file. The `reference-descriptor` stanza may contain one or more `resource-factory` or EJB reference mappings. This EJB needs to map a `jdbc/MovieInfoDataSource` to a JDBC TX data source that you will later define as `MovieInfoDataSource` by using the WebLogic Administration Console. You'll also need to have a `<jndi-name>` entry for your bean to bind it to WebLogic Server's global JNDI tree.

Although the Java Smart Ticket application was designed to run with Cloudscape, very little effort is needed to modify the BMP bean classes to get them to run with PointBase 4.2 server. In the original `CustomerEJB` bean class, the SQL statement to find a customer uses parentheses in the join clause. PointBase doesn't support this. You can code around that; I did so by rewriting the

dbFindCustomer method to drop the join (see Listing 3).

When all the EJB descriptors are done, you must jar and compile the EJBs by modifying the %SMARTICKET_HOME%\build.xml. In the %SMARTICKET_HOME%\build.xml file, I added the Ant target weblogic.compile.ejb to supercede j2ee.compile.ejb and j2ee.package.ejb (see Listing 4). What I would like to highlight is shown below:

```
<target name="weblogic.compile.ejb">
...
<copy file="${src.dir}/ejb-jar.xml" tofile="${smarticket.buildjardir}/META-INF/ejb-jar.xml"/>
<copy file="${src.dir}/weblogic-ejb-jar.xml"
tofile="${smarticket.buildjardir}/META-INF/weblogic-
ejb-jar.xml"/>
<jar jarfile="${smarticket.ejbjar}"
basedir="${smarticket.buildjardir}"
update="yes"/>
<java classname="weblogic.ejbc"
fork="yes"
classpath="${java.class.path}">
<arg line="-compiler javac -keepgenerat-
ed ${smarticket.ejbjar}"/>
</java>
...
</target>
```

The first thing this does is copy the weblogic-ejb-jar.xml file to the \${smarticket.buildjardir}/META-INF directory where the ejb-jar.xml file is. Next, instead of just simply JARring up your compiled classes and descriptors and deploying the resulting JAR, run weblogic.ejbc utility on the JAR. The output JAR, ejb_st.JAR, is symbolized by \${smarticket.ejbjar}.

Although not required, running weblogic.ejbc before deployment yields several benefits. The most immediate advantage is that it provides validation on the EJBs according to the EJB specification. Errors in the EJB class files and descriptors can be detected and fixed before the EJB is deployed to a WebLogic server. Another benefit is that EJB compiling and stub generation are decoupled from server startup.

Smart compilation is a new feature of the weblogic.ejbc utility in WebLogic Server 7.0. In previous versions, an input and output JAR were both required as arguments to the weblogic.ejbc utility. This is no longer true. Instead, you can simply pass one JAR to the weblogic.ejbc utility. The next time the weblogic.ejbc utility is run on the JAR file, it can determine whether or not the bean classes within the JAR file need to be regenerated.

To use smart compilation, remove the Sun pre-built %SMARTICKET_HOME%\bin\ejb_st.jar file, but at the same time disallow immediate deletion of the ejb_st.jar file when we build it for the first time. I commented out the second delete in this Ant target in the %SMARTICKET_HOME%\build.xml file (Listing 4):

```
<target name="j2ee.clean">
<delete dir="${j2ee.build.dir}" />
<!-- <delete dir="${j2ee.dist.dir}" /> -->
</target>
```

We will not build the EJBs just yet, as a few more changes are required in the %SMARTICKET_HOME%\build.xml file. We will go over these changes in later on.

How to Port the Web Application

Porting the Web application component of the Java Smart Ticket application is very similar to porting the EJB component. In addition to having the Java Smart Ticket web.xml file, we need to create a corresponding weblogic.xml file.

Whereas the weblogic-ejb-jar.xml file is required for all EJBs and weblogic-cmp-rdbms-jar.xml files for CMP EJBs, the weblogic.xml file is needed only if you have attributes to set. It allows you to map security names to a security realm and to map resources to JNDI. It also allows you to define JSP, session, container, and character set parameters.

The weblogic.xml file will make two types of references in the reference-descriptor stanza: resource factory and EJB. Like the weblogic-ejb-jar.xml file, we need to map the jdbc/MovieInfoDataSource to the MovieInfoDataSource transactional datasource. After all the resource-description elements are defined, we then define the EJB references that the Web application needs. The ejb-ref-name, ejb/MovieInfo, is mapped to Smarticket.MovieInfoEJB. As you know, this is the JNDI name I had assigned the MovieInfo EJB in the weblogic-ejb-jar.xml file. Here is what it would look like in the weblogic.xml file (see Listing 5):

```
<reference-descriptor>
<resource-description>
<res-ref-name>jdbc/MovieInfoDataSource</res-ref-
name>
<jndi-name>MovieInfoDataSource</jndi-name>
</resource-description>
...
<ejb-reference-description>
<ejb-ref-name>ejb/MovieInfo</ejb-ref-name>
<jndi-name>Smarticket.MovieInfoEJB</jndi-name>
</ejb-reference-description>
...
</reference-descriptor>
```

BEA eWorld Europe

www.bea-eworld.com

And to the %SMARTICKET_HOME%\build.xml file, I add the following to j2ee.web.package ANT target (see Listing 4):

```
<copy todir="${j2ee.build.web.dir}/WEB-INF"
      file="${src.dir}/weblogic.xml"
      overwrite="true" />
```

I placed this copy stanza right after the stanza that copies the web.xml file to \${j2ee.build.web.dir}\WEB-INF

Putting it All Together

BUILDING THE SMARTICKET.EAR

The Sun J2EE Reference Implementation server isn't needed to build, deploy, or run the ported Java Smart Ticket application.

To properly set up your environment to build the Java Smart Ticket application for WebLogic Server 7.0, you'll need to do several things. In a CMD window, navigate to %SAMPLES_HOME%\server\config\examples and run setExamples-Env.cmd. This sets up your WebLogic Server development environment. Next, set the J2MEWTK_HOME environment variable equal to the root directory where you installed the J2ME toolkit. For your convenience, I have included setenv.cmd to set J2MEWTK_HOME equal to C:\j2mewtk, the default installation directory.

In the %SMART_TICKET%\build.xml file make sure that this Ant target

```
<target name="j2ee" depends="j2ee.compile.ejb,
j2ee.compile.web,j2ee.package.ejb,j2ee.package.web,j2ee
.package" />
```

is changed to

```
<target name="j2ee"
depends="weblogic.compile.ejb,j2ee.compile.web
,j2ee.package.web,weblogic.package" />
```

to call the new Ant targets that we need: one we discussed in "How to Port the Enterprise Java Beans"; the other, I'll discuss here.

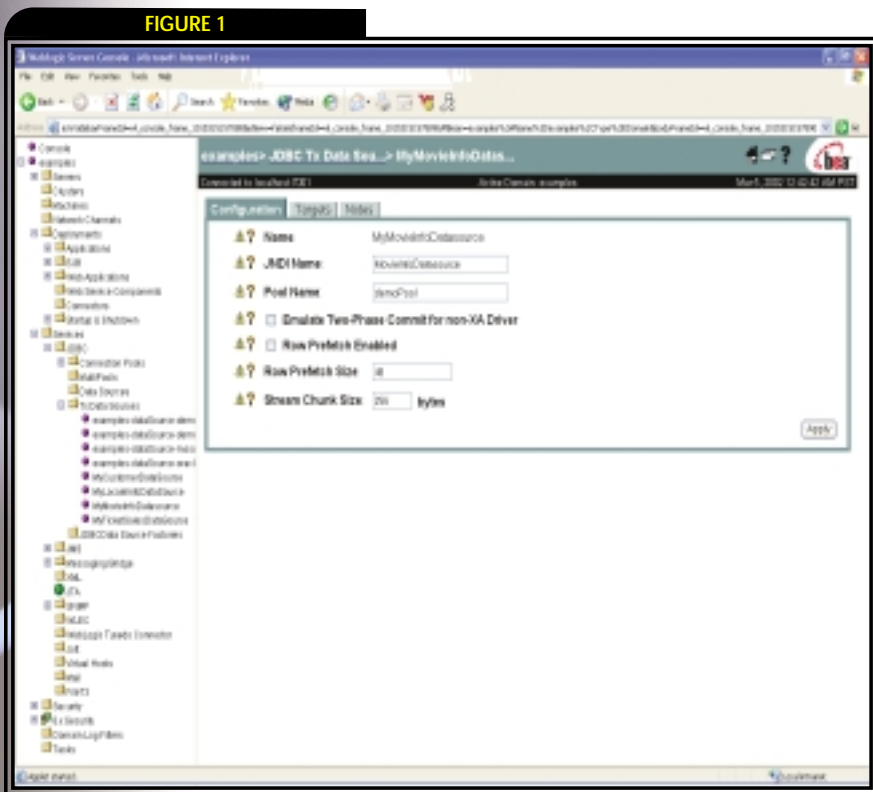
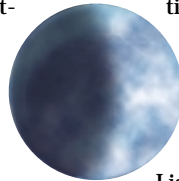
WebLogic server doesn't require its applications to be packaged by the Sun J2EE Reference Implementation tools. This has two consequences for us. First, we must create an application.xml file by hand; I saved mine under %SMARTICKET_HOME%\src (see Listing 6). It's important, as it's the deployment descriptor for the smartticket.ear. Second, I have added a weblogic.package Ant target to supercede j2ee.package. All this target does is JAR up ejb_st.jar, web_st.jar, and META-INF/application.xml into smartticket.ear.

I've added the weblogic.package Ant target to supercede the j2ee.package. This target will JAR up ejb_st.JAR, web_st.jar, and META_INF/application.xml into smartticket.ear. I've made these changes because we're not using the Sun J2EE Reference Implementation tools for packaging. Consequently, we also need to create an application.xml file; this file will be the deployment descriptor for the smartticket.ear. I saved mine under %SMARTICKET_HOME%\src (see Listing 6).

UPLOADING DATA TO THE POINTBASE DATABASE

As you may know, the J2EE Reference Implementation server ships with the Cloudscape RDBMS. By default, the %SMARTICKET_HOME%\populate.bat script uploads the %SMARTICKET_HOME%\src\smartticket.sql ddl to a Cloudscape database. To do the same for the PointBase 4.2 Server, the evaluation RDBMS included with WebLogic Server 7.0, we need to make sure that smartticket.sql and populate.bat are PointBase friendly. The only change you'll need to make to smartticket.sql is to change the 10 "int" data types to "integer." (I've provided a smartticketPointBase.sql with the changes for your convenience. Next, update the %SMARTICKET_HOME%\populate.bat script to use the PointBase driver with the following:

```
set POINTBASEHOME=%SAMPLES_HOME%\server\eval\pointbase
java utils.Schema
jdbc:pointbase:server://localhost/demo,database.home=%P
OINTBASEHOME% com.pointbase.jdbc.jdbcUniversalDriver -u
examples -p examples -verbose
./src/smartticketPointBase.sql
```



Creating a data source

PwC

www.pwcconsulting.com/nwn

Starting the WebLogic 7.0 Examples server will automatically start the PointBase 4.2 server in a minimized CMD window. You can start the WebLogic 7.0 Examples server by selecting the Start Menu item, "Start Examples Server", in the SERVER EXAMPLES folder or by open-

ing a CMD window and executing %SAMPLES_HOME%\server\config\examples\start ExamplesEnv.cmd.

In another CMD window, source your environment by running %SAMPLES_HOME%\server\config\examples\setExamplesEnv.cmd.

Run %SMARTICKET_HOME%\populate.bat. Your database should now be set up.

DEPLOY AND CONFIGURE THE APPLICATION

Before installing the Java Smart Ticket application, we need to configure the JDBC Tx Data Sources that it will need.

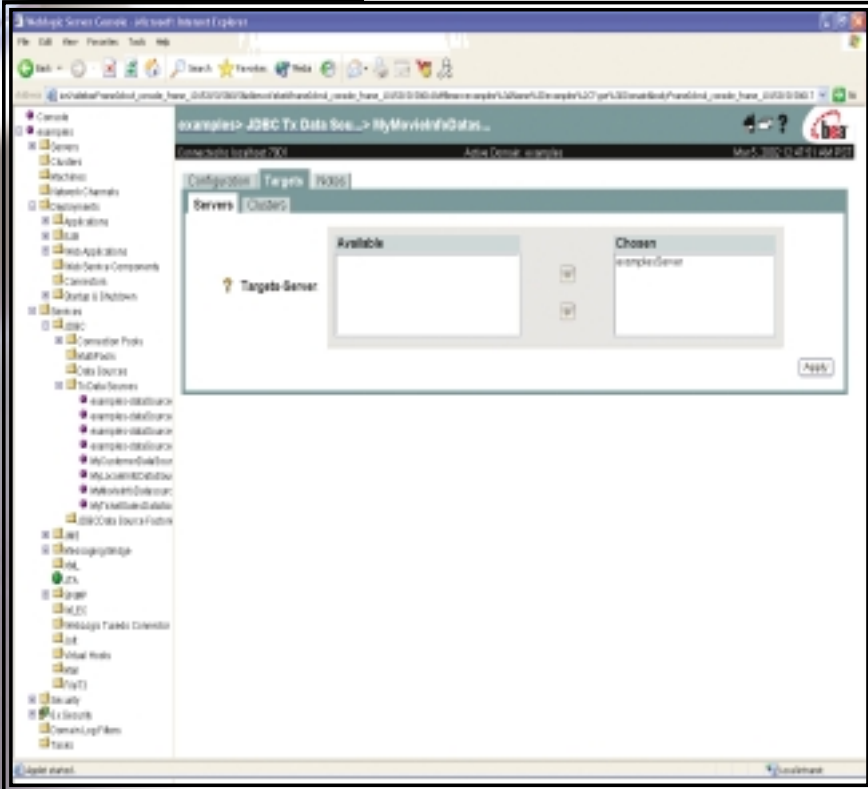
1. Start the Examples server if you have not already. Browse to <http://localhost:7001/console> and sign in to the WebLogic Administrative console
2. In the "Service Configurations" section and under the "JDBC" heading, choose "Tx Data Sources."
3. Next, click "Configure a new JDBC Tx Data Source..." and go to the Configuration Tab of a new JDBC Tx Data Source. I'll create a MovieInfoDataSource as an example (see Figure 1).
4. Enter a name for the data source; I call it "MyMovieInfoDataSource".
5. Enter "MovieInfoDataSource" in the JNDI field. Remember, this corresponds to the JNDI references I made in our EJB and Web application descriptors.
6. For the connection pool, enter "demoPool". This corresponds to the connection Pool that ships with the Examples Server. (Verify this by going to the Services\JDBC\Connection Pool node on the left-hand side.)
7. Leave everything else as default and click "Create".
8. Click the Targets Tab.
9. Highlight the examplesServer in the Available column and click the right arrow to move it to the "Chosen" column (see Figure 2).
10. Be sure to click "Apply".

You have successfully configured a JDBC Tx Data Source. Do the same for the three other data sources: TicketSalesDataSource, CustomerDataSource, and LocaleInfoDataSource.

The final task is to configure, or install, the application.

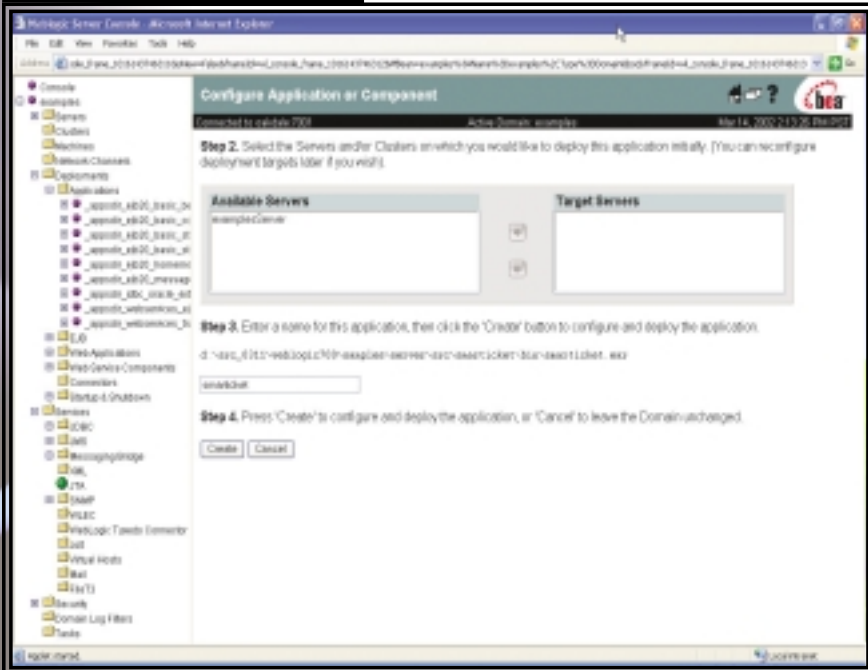
1. In the left pane of the Administration Console, browse to the examples\Deployments\Applications node.
2. Click "Configure a new Application..." and browse to %SMARTICKET_HOME%\bin and select "smartticket.ear" (see Figure 3).
3. Highlight the examplesServer in the "Available Servers" column and click the right arrow to move it to the "Target Servers" column.
4. Click "Apply".
5. Click "Create" and a Configuration tab will appear for the application.

FIGURE 2



Targeting the data source

FIGURE 3



Targeting the application

THE LARGEST INTERNATIONAL

WEB SERVICES CONFERENCE & EXPO IN THE WORLD!

WIN A \$35,000 LUXURY CAR!



ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

WEB SERVICES SKILLS, STRATEGY, AND VISION

REGISTER ONLINE TODAY

**FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!**

VISIT WWW.SYS-CON.COM

Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at Web Services Edge East 2002. Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Roundtable panel that will debate and discuss the realities and promise of Web Services.



A Sampling of Web Services-Focused Sessions

- PRACTICAL EXPERIENCES WITH WEB SERVICES AND J2EE
- STATE OF THE WEB SERVICES INDUSTRY
- THE ODD COUPLE: MAKING .NET AND J2EE WORK TOGETHER
- EXPLORING THE .NET MY SERVICES INITIATIVE
- STANDARDS WATCH
- GUARDING THE CASTLE: SECURITY & WEB SERVICES



STAN RUDDY
CONFERENCE TECH CHAIR
WEB SERVICES TRACK CHAIR
EDITOR-IN-CHIEF
WEB SERVICES JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 J-TECHNOLOGY PROFESSIONALS

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- J-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...



JEAN-FRANCOIS ARAMATIC
SENIOR VP R&D, ILOG
FORMER CHAIRMAN, W3C



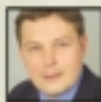
DAVE CHAPPELL
VP CHIEF TECHNOLOGY EVANGELIST
SONIC SOFTWARE



GREGG KIESSLING
CEO
SITRAXA



ANNE THOMAS MANES
CTO
SYSTEMET



BARRY MORRIS
CEO
IONA



ERIC RUDDER
SENIOR VP DEVELOPER
AND PLATFORM EVANGELIST
MICROSOFT



PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit Information

CONTACT: MICHAEL PESICK
115 CHESTNUT RIDGE RD.
MONTVALE, NJ 07045
201-980-3357
MICHEL@SYS-CON.COM

web services **EDGE**
conference & expo

JUNE 24-27
JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

SPONSORED BY:



MEDIA SPONSORS

Federal Computer Week

WebServices.org

XML TIMES.com

Java Skyline

GF Advisor

WebServicesMail

WIRE

XML.org



SDTimes

JAVA SOURCE

wireless

XML SOURCE

WebLogic

WebServices

WebSphere

Colofusion.com

OWNED AND PRODUCED BY



Simplex Knowledge Company

www.skc.com/J2EE



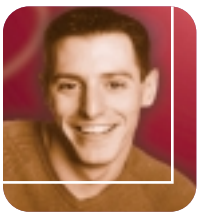
Architecting and Architecture

I was goofing off this weekend, trying to figure out what would be the best topic to write about for this month's architecture column. And, like any good columnist, I procrastinated until Sunday night (the article was due on Monday morning).

Understanding the WebLogic Workshop Architecture

A NEW CONCEPT IN A FAMILIAR STYLE

BY TYLER JEWELL



AUTHOR BIO...

Tyler Jewell is BEA's director of technology evangelism. He's the author of *Java Web Services* and the coauthor of *Mastering Enterprise JavaBeans 2.0* and *Professional Java Server Programming (J2EE 1.3)*. He writes widely on the Web on the subject of both J2EE and Web services.

CONTACT...

tyler@bea.com

Right on time! While sitting at my computer, I couldn't help but wander off to random Web sites and all the while I was instant messaging a complete stranger, Andrea. In my angst to write a great column, I turned to Andrea and asked her what this month's topic should be. She responded with, "flying monkeys and plastic brains."

Hmmm. Although an appealing topic to write about, I was concerned about how appropriate it would be for this audience. And, even though I can't write about monkeys and brains, there is a metaphor here: understanding flying monkeys and plastic brains is like WebLogic Workshop's architecture – everyone is initially confused by it.

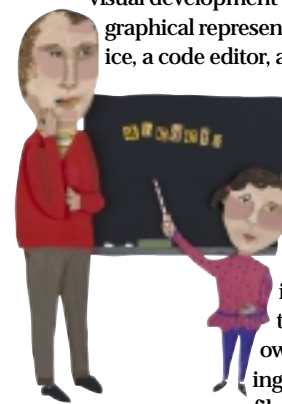
BEA has invested a lot of marketing and engineering effort into WebLogic Workshop. It was launched with a bang at our eWorld conference in February. I even had the pleasure of doing a launch demo as part of Alfred Chuang's keynote. But, despite all this launch hysteria, there hasn't been much information about Workshop's architecture. This column dives into this.

High-Level Architecture

Figure 1 details the high-level architecture of what WebLogic Workshop has to offer. There are

two portions: a runtime framework and a visual development environment. The value and complexity is located in the runtime framework. At the core of WebLogic Workshop is a Web service definition. The Web service definition is located in a JWS file that is a pure Java file residing in WebLogic Server. The runtime framework manages JWS files and converts them to J2EE applications at appropriate times. Additionally, the runtime framework provides debugging services to the development environment. And, it adds Web services support by providing SOAP protocol and WSDL mapping services.

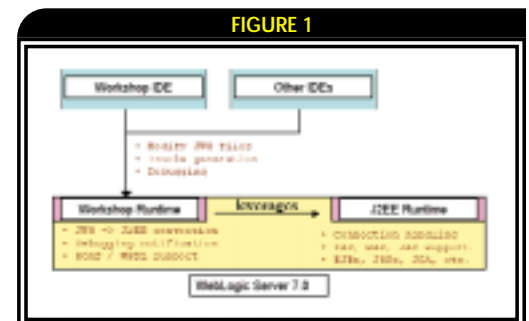
The Workshop IDE is a visual development environment developers can use to create JWS files. The visual development environment provides a graphical representation for the Web service, a code editor, and the ability to interface with the runtime framework. Since the runtime framework is open and extensible, other IDEs, such as WebGain Studio and Borland JBuilder, can interact with the runtime and provide their own structure for managing Web services and JWS files.



Is This Really New?

The WebLogic Workshop architecture isn't actually a new approach to application deployment. Even though the JWS file format is a new concept, the deployment model is very familiar. The JWS deployment model is very similar to the JSP deployment model. Figure 2 provides more information about this process.

At the core of the architecture, a JWS file is cre-



Overall Structure

Web Services JOURNAL

Special Introductory Offer
SAVE \$13.89*

The world's leading independent Web Services information resource

Get Up to Speed with the Fourth Wave in Software Development

- Real-World Web Services: Is It Really XML's Killer App?
- Demystifying ebXML: A Plain-English Introduction
- Authentication, Authorization and Auditing: Securing Web Services
- Wireless: Enable Your WAP Projects for Web Services
- The Web Services Marketplace: An Overview of Tools, Engines and Servers
- Building Wireless Applications with Web Services
- How to Develop and Market Your Web Services
- Integrating XML in a Web Services Environment
- Real-World UDDI
- WSDL: Definitions and Network Endpoints
- Implementing SOAP-Compliant Apps
- Deploying EJBs in a Web Services Environment
- Swing-Compliant Web Services
- and much, much more!

Only \$69.99 for 1 year (12 issues)
Newsstand price \$83.88 for 1 year



SYS-CON MEDIA

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 Web services.

*Offer subject to change without notice

SUBSCRIBE AND SAVE

XML JOURNAL

Offer subject to change without notice

ANNUAL NEWSSTAND RATE	
\$93.88	
YOU PAY	
\$77.99	
YOU SAVE	
\$5.89	Off the Newsstand Rate

DON'T MISS AN ISSUE!

Receive 12 issues of *XML-Journal* for only \$77.99! That's a savings of \$5.89 off the annual newsstand rate.

Sign up online at www.sys-con.com or call 1 800 513-7111 and subscribe today!

In May XML-J:

XSLT on Wall Street

Sam Natarajan shows how a Wall Street firm utilized XSLT for its financial applications.

Data Driven Web Graphics with SVG

Pramod Jain and Satya Komatineni present a declarative approach to creating a middle tier for SVG applications.

XML Technologies in the Presentation Layer

Mark Spears provides background on Web technologies.

Create Sortable HTML Tables Using XSLT

Sean McMullan explains the trick to browser-side transformations: get both the XML data and XSLT presentation logic into the browser, where it can be transformed via JavaScript.



ated. A JWS file is a Java file with a standardized set of Javadoc tags that are used to mark up attributes, methods, and inner classes. JWS files are placed into a Web application directory on a J2EE server that supports JWS. This would be the same location where JSPs are located.

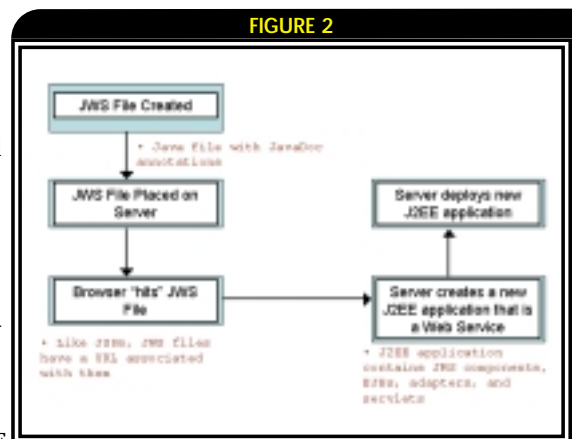
Similar to JSPs, each JWS file can be referenced through a unique URL. For example, the URL may look like `http://server_name:port/application_name/JWSFile.jws`. Any program that can generate the right protocol requests to this URL can then invoke a message for this URL. This could be a browser, directly from the Workshop IDE, or from another IDE. When a URL request is received by the server, the server parses the JWS file and creates a new J2EE application that consists of EJBs, JMS destinations, JCA adapters, and servlets. These files are packaged into an EAR file and deployed into the same server as a new application. This new application is exposed as a Web service and immediately made available to external clients.

What About Control Files?

WebLogic Workshop control files are used to integrate with enterprise resources. In this first version of Workshop, there are timer, EJB, JMS, JCA (AppView), and service controls. A control file is similar to a JWS file: it's a Java interface that contains Javadoc annotations for indicating how the integration should occur.

When a JWS file is created, it contains a reference to a control file for each instance of a resource that is going to be used in the implementation. When a JWS file is initially converted into a J2EE application, the JWS translator parses the JWS file and discovers all of the control dependencies for that application. The translator then takes each control file and creates an implementation for that control. For example, for an EJB control file, the translator would create a Java proxy from the JWS file to the EJB. For the timer control file, the translator would set up a timer queue with a message listener to receive timing messages. For all of the controls that are used by a single JWS file, their implementations are bundled as part of the application created for the broader JWS file.

So, whereas a single JWS file will get converted into a complete J2EE application, a single control file will be converted into an implementation that is hosted as part of a J2EE application defined by a JWS file.



IWS Deployment

Conclusion

In a nutshell, that's the core of the WebLogic Workshop architecture and how the runtime engine works. There are other aspects that the runtime engine provides to WebLogic Workshop, such as low-level Web services protocol support, data type encoding, XMAP translation, and others. Also, IDEs can add on a variety of value-add services as well: configuration management support, support for adding controls fluidly, support for integrating other Web services (how do you locate and import these things?), different graphical representations of JWS files, and so on. If you haven't played with Workshop yet, I suggest you try it. You can download it today at <http://com.merce.beasys.com>.

Now in More than 5,000 bookstores worldwide

subscribe **Now!**

FOR FAST DELIVERY

Go Online and Subscribe Today!



Helping you enable intercompany collaboration on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks and more!

SPECIAL INTRODUCTORY OFFER
SAVE \$31*
 HURRY, DON'T DELAY! OFFER EXPIRES JUNE 30, 2002

WebLogicDevelopersJournal.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 WebLogic. *Only \$149 for 1 year (12 issues) regular price \$180.



WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
BEA	www.developer.bea.com	408-570-8000	2,3
BEA eWorld Europe	www.bea-eworld.com	404-240-5506	25
EnginData	www.engindata.com		38
JDJ Edge	www.sys-con.com	201-802-3069	53
Mongoose	www.mongooseotech.com	281-461-0099	11
Performant	www.performant.com	866-773-6268	39
PBDJ	www.powerbuilderjournal.com	800-513-7111	19
Precise	www.precise.com/wldj	800-310-4777	7
PwC	www.pwiconsulting.com/nwn	267-330-6355	27
ReportMill	www.reportmill.com	214-513-1636	21
Simplex Knowledge Company	www.sk.com/J2EE	845-620-3700	31
Simplex Knowledge Company	www.sk.com/training	845-620-3700	55
Sitraka	www.sitraka.com/jprobe/wldj	800-663-4723	15
Sitraka	www.sitraka.com/jclass/wldj	800-663-4723	9
Sitraka	www.sitraka.com/performance/wldj	800-663-4723	60
Sun Microsystems	www.sun.com/servers/entry/beapromo3	800-765-9200	37
SYS-CON Custom Media	www.sys-con.com	201-802-3022	49
SYS-CON Reprints	www.sys-con.com	201-802-3026	30
Thought, Inc.	www.thoughtinc.com	415-836-9199	59
WebLogic Developer's Journal	www.sys-con.com	800-513-7111	35
Web Services Edge	www.sys-con.com	201-802-3069	29, 56, 57
Web Services Journal	www.wsj2.com	800-513-7111	33
Web Services Resource CD	www.jdstore.com	201-802-3012	40, 41
Wily Technology	www.wilytech.com	888-GET-WILY	4
XML Edge	www.sys-con.com	201-802-3069	43
XML Journal	www.sys-con.com	800-513-7111	34

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser.

NEXT MONTH

Building Service-Oriented Collaborative Business Applications

The implementation challenges and infrastructure-level requirements

Consuming WebLogic 6.1 Web Services with ASP.NET

Planning for the eventual removal of adapter code

Security Integration with BEA WebLogic

Seamlessly integrate WebLogic J2EE components with other third-party security services

Mongoose Technologies' Erick Rivas Talks About Integration

Plus, More From
Tyler Jewell, Peter Holditch, Sam Pullara, and Dave Cooke

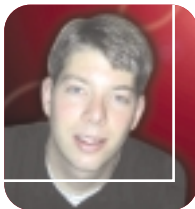


Every developer has experienced it. The application that ran so well in testing hangs or performs miserably under load.

Common WebLogic Server Deadlocks and How To Avoid Them

AVOID COMMON APPLICATION MISTAKES - AND LET YOUR THREADS PLAY OUT

BY ROB WOOLLEN



AUTHOR BIO...

Rob Woollen is a senior software engineer on the WebLogic Server development team at BEA Systems. He holds a bachelor's degree in computer science from Princeton University.

CONTACT...

rwoollen@bea.com

While there are many possible causes of performance degradation or hangs, this article can't possibly cover them all. Instead, we'll look at three common mistakes in WebLogic Server applications that can deadlock the server or bring your performance to a screeching halt.

Thread Dumps

The best Java tool for diagnosing deadlocks is a Java virtual machine thread dump. A thread dump is a snapshot of the virtual machine's current state, including stack traces for each Java thread. Many virtual machines also include information about the Java monitors held by each thread. Monitor information is especially useful for diagnosing deadlocks and performance problems in your application. On Windows platforms, you can generate a thread dump by pressing Ctrl-Break in the virtual machine's window. On UNIX systems, a SIGQUIT signal must be sent to the Java virtual machine process. This can be done with a kill -3 <process id>.

Deadly Embrace

The classic deadlock problem is the deadly embrace: Thread 1 owns Lock A and waits on Lock B, Thread 2 owns Lock B and waits on Lock A. These threads are deadlocked and will remain blocked in this state. In many cases, the remain-

ing threads will eventually enter the deadlock by attempting to acquire Lock A or Lock B and waiting. For instance, you might have a servlet that calls a synchronized method on the B object. If B's monitor is already held in a deadlock, any subsequent servlet request that attempts to acquire that monitor will enter the deadlock.

A thread dump is one of the best ways to discover a deadly embrace deadlock. Most virtual machines include a thread state for each Java thread in the dump. The most common thread states are: R - running; MW - monitor wait; CW - condition wait. Threads in the MW state are blocked, waiting to enter a synchronized block and acquire a Java monitor. Since the thread dump includes the Java thread's stack trace, it's also possible to determine which monitor is blocking the thread. If multiple threads are in the MW state on the same monitor, it's a good indication that there's either a lot of contention for this monitor, or the server is deadlocked. In a deadlock situation, you should be able to determine the other threads blocked in MW and their held monitors.

There are two classic techniques for solving deadly embrace deadlocks: deadlock avoidance and deadlock detection. Deadlock avoidance is merely changing or structuring your code so that it can't hit the deadlock case. A common solution is to implement lock ordering. If Lock A is always acquired before Lock B then you can't have a deadly embrace on these two locks. It's also a good idea to minimize the time that a monitor is held. Every extra line of code that runs under a monitor is another chance for someone to add a deadlock. Deadlock detection is commonly implemented by databases for their locks, but it doesn't usually find its way into programming languages. In deadlock detection, deadlocks are automatically discovered and one or more deadlock participants (known as the victims) are killed and release their locks to break the deadlock. Java virtual machines do not break deadlocks on Java monitors, so deadlock avoidance is necessary.

Out-of-Threads Deadlock

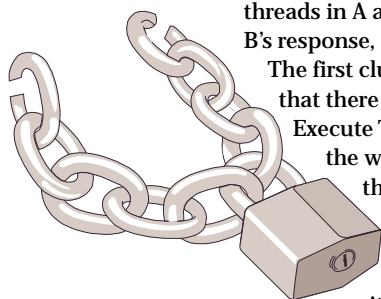
Another common way to deadlock an application is what I call the "out of threads" deadlock. Unfortunately, this deadlock often doesn't show up until a load test or, in the worst case, when your production application receives a lot of traffic. In this scenario, your WebLogic Server is running with a fixed number of threads. The applica-

Sun Microsystems

www.sun.com/servers/entry/beapromo3

tion includes logic where a given request or action performs work in one thread and then blocks on work that must be done in another thread.

For instance, an application might open an HTTP socket connection to its own server instance, post a request, and wait for a response. If all threads post at the same time, there are no available threads to generate a response. Another common scenario is when Server A makes an RMI call to Server B and blocks waiting for a response. Server B then calls back into A before it generates the response to A's initial call. If all threads in A are exhausted then A is waiting for B's response, and B is waiting for A's response.



The first clue in an "out of threads" dump is that there are no idle WebLogic Server Execute Threads. The Execute Threads are the worker threads in the server that run the application code. An idle ExecuteThread will be in condition wait (CW) in a method named ExecuteThread.-waitForRequest. This thread is available and waiting for incoming work. In an "out of threads" deadlock, you won't see any idle threads, and all other worker threads will be blocked, usually waiting for a response.

Unfortunately, determining that there are no idle threads is slightly more complicated than just searching for ExecuteThread.waitForRequest. WebLogic Server uses thread pools internally to avoid having to continually create new threads. It also offers multiple thread pools and uses several different thread pools within the server. Thread pools would be a good topic for an entire article so we won't cover them in detail here, but it's important to understand for this topic that most user work is performed on the "default" queue. In an "out of threads" deadlock, it's important to note whether there are no idle threads on the "default" queue.

The best way to avoid "out of threads" deadlocks is to analyze your architecture and remove the common mistakes that produce these deadlocks. In particular, never open a socket connection to your own server instance. Also, avoid synchronous request/response APIs that include callbacks. In general, asynchronous communication works well for server-to-server or application-to-application calls. Both messaging (JMS) and Web services include asynchronous support. One of the advantages of asynchronous communication is that the calling thread is not blocked waiting for the response.

EnginData Presents

2002 Developer Market Survey Reports



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

Preview and order reports at www.engindata.com

engindata.com

engindata ✓
RESEARCH



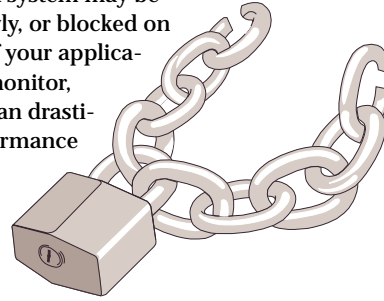
Monitors and External Systems

In addition to diagnosing deadlocks, thread dumps can be useful for analyzing performance problems. One architectural mistake that is often evident from thread dumps is holding Java monitors while making external requests. For instance, an application might enter a synchronized block on its business data object and then log some information on an external system. In the common case, this might work fine if the logging to the external system is running relatively quickly. However, it really runs into trouble under load. In a production system, the external system is probably slower, and there are many concurrent requests trying to get access to the monitor. Usually, the entire system slows down dramatically and essentially becomes single-threaded.

The “single-threaded” problem shows up in thread dumps when there are many threads that are blocked in MW for the same monitor. There is then a single thread that already owns the monitor and is often blocked waiting for some external system to return. For instance, it’s often blocked waiting for a network socket read to return. In thread dumps that display the current monitor owners, it’s obvious which thread holds the contested monitor. In less verbose thread dumps,

you’ll have to analyze the stack trace for each thread and determine which monitors it’s holding. Since a monitor is only ever owned by one thread, this is not as bad as it initially sounds.

The “single-threaded” problem can be avoided by never holding Java monitors while making requests to external systems. Fundamentally, holding monitors and running someone else’s code is inviting deadlock or extremely poor performance. The external system may be down, performing slowly, or blocked on some other resource. If your application is holding a Java monitor, these external events can drastically reduce your performance to a crawl.



Conclusion

This article demonstrates some common WebLogic Server application mistakes and how they contribute to deadlocks or poor performance. Thread dumps are invaluable in debugging these types of problems. Hopefully, the techniques in this article will help you find your way out the next time your application becomes unresponsive. ●

Performant
www.performant.com

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN

400

EXCLUSIVE

WEB SERVICES
& XML
ARTICLES



EDITED BY
SEAN RHODY



\$119
CD
VALUE
FROM
WEB SERVICES
JOURNAL

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

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.

Easy-to-navigate HTML format!

Bonus:

Full .PDF versions of every WSJ & XML-J published
since the first issue

XML in Transit	XML Industry	XML &	UML
XML B2B	Insider	Databases	Integration
Java & XML	<e-BizML>	Electronic Data	WSDL
The XML Files	XML & Business	Interchange	Beginning
XML & WML	XML Demystified	Ubiquitous	Web Services
Voice XML	XML &	Computing	Web Services
SYS-CON Radio	E-Commerce	Information	Tips & Techniques
XML & XSLT	XML Middleware	Management	Frameworks
XML & XSL	XML Modeling	Objects & XML	Management
XML & XHTML	CORBA & XML	XML Pros & Cons	Security
2B or Not 2B	Data Transition	Java Servlets	UDDI
XML Script	XML @ Work	XML Filter	.NET

3
YEARS
25
ISSUES
400
ARTICLES
ONE CD



Special Limited Time Price

Now
Shipping

\$79

+ S&H

ONLINE
ORDER AT
JDJSTORE.COM

SAVE
\$40

www.JDJSTORE.com

OFFER EXPIRES JUNE 30, 2002



Are you looking for something to differentiate yourself from your peers in this tighter job market? BEA WebLogic Server Certification may be for you. It provides employers, or potential employers, with additional evidence that you're qualified for developing solutions on the BEA WebLogic Server platform.

gests that you have at least a "working knowledge of Java" before taking the test. In addition, BEA suggests that you have three to six months of hands-on WebLogic Server experience as a prerequisite. Though it is possible to pass by simply studying the appropriate material, having some real experience with WebLogic Server development will certainly make it easier.

BEA-recommended prerequisites include two BEA education classes: "Developing Applications using BEA WebLogic Server Version 6.0" and the "J2EE Core Technologies and Developing Enterprise Applications with Enterprise JavaBeans Using WebLogic Server Version 6.0." Although these classes don't take the direct approach to helping you gain your developer certification, they do provide a wealth of study material. To get the full benefits from the education services to cover the test, though, you really need to take an additional class on administering WebLogic Server: "BEA WebLogic Server Version 6.0 Administration". For more information on the BEA education classes, visit http://education.bea.com/education/sales/course_catalog.jsp.

There's another prerequisite that isn't listed on the BEA Web site: this series of articles. I'll provide a helpful study guide that will enable any developer familiar with BEA WebLogic Server 6.0 development, or in some cases, a developer just familiar with general J2EE development, with enough material to effectively study for the test.

Signing Up

Let's start by signing up to take the test. First, you need to visit the Prometric Web site, www.2test.com. Select "Find a Test Center" from the top navigation bar. Once you're at the "Test Center Locator," enter BEA Systems and your country. Make sure you have test number "0B0-200 WebLogic Server 6.0 Developer Certification Exam" and your state. After you enter your language, find the test center that's most convenient to you. Make a note of its name and address. In addition, be sure to record the number of the test center, which is usually the two-digit state code followed by a number. Now you're ready to call Prometric to schedule the test.

You can reach Prometric at 877-205-6873. Simply tell the operator which test you wish to take, 0B0-200, and the number of the test center you found on the Web site. The operator will offer

Getting Started

BECOMING A BEA CERTIFIED DEVELOPER ON WEBLOGIC SERVER 6.0

BY DAVE COOKE



AUTHOR BIO

David Cooke is an experienced software developer, currently working for Ness Technologies, Inc. (www.ness-usa.com), a consulting firm located in Dulles, VA. In his current position, he utilizes Java and BEA WebLogic Server 6.0 to build J2EE-compliant e-commerce systems for a variety of clients. Dave maintains Microsoft, Java, and BEA developer certifications.

CONTACT...

dave.cooke@ness-usa.com

Are you looking to advance in your professional career, or get a raise? BEA WebLogic Server Certification may be the thing for you! Many employers offer salary increases or bonuses for attaining technical certifications.

One such certification program is jCert, created by BEA Systems, Hewlett-Packard, IBM, Oracle, Sun Microsystems, and Sybase. The BEA Certified Developer Test qualifies as a jCert Initiative test. More information on this can be found at www.jcert.org.

There are many reasons to become a BEA Certified Developer, but the best is to prove to yourself that you are truly qualified.

Certification starts with the BEA WebLogic Server 6.0 Certification Test. This is a multiple-choice/short answer exam that tests your knowledge of WebLogic Server 6.0. By passing it you demonstrate a level of knowledge that qualifies you as a BEA Certified Developer. In addition, passing the test allows you to display the BEA Certified Developer logo on your resume and/or business cards.

BEA recently changed the prerequisites for this test. Previously, you needed to have your Sun Java Programmer Certification before you could qualify to take the WebLogic Server Developer Certification Test. However, BEA has eased the requirements and now their Web site simply sug-

THE LARGEST INTERNATIONAL

XML

CONFERENCE & EXPO IN THE WORLD!

**WIN A
\$35,000
LUXURY CAR!**



ATTENDEES WILL BE INVITED TO TAKE A GOLF SHOT TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

XML-NEXT G OF ENTERPRISE DEPLOYMENT

REGISTER ONLINE TODAY

**FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!**

VISIT WWW.SYS-CON.COM

Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

Focus on XML during information-packed days while you enjoy an XML/Web Services keynote panel, comprehensive conference sessions, and an unparalleled opportunity to exchange ideas with peers and industry leaders.



A Sampling of XML-Focused Sessions

- XML STANDARDS - AN OVERVIEW
- OASIS STANDARDS UPDATE
- UBL - A UNIVERSAL BUSINESS LANGUAGE
- BRINGING XML TO PKI
- LINK MANAGEMENT WITH XLINK
- PRACTICAL XSLT AND XPATH
- ENTERPRISE CONTENT MANAGEMENT WITH XML
- XML IN THE ENTERPRISE AND INTER-ENTERPRISE WORLD



NORBERTO MIKELA
XML CHAIR
BOARD OF DIRECTORS, OASIS
INDUSTRY EDITOR
XML JOURNAL

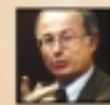
Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 J-TECHNOLOGY PROFESSIONALS

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- J-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...



JEAN-FRANCOIS ABRAMATIC
SENIOR VP R&D, ILOG
FORMER CHAIRMAN, W3C



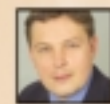
TYLER JEWELL
PRINCIPAL TECH
EVANGELIST • BEA



DAVID LITWACK
CEO
SILVERSTREAM



ANNE THOMAS MANES
CTO
SYSTEMT



BARRY MORRIS
CEO
IONA



RICK ROSS
FOUNDER
JAVALOBBY



PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit Information

CONTACT: RICHARD ANDERSON
115 CHESTNUT RIDGE RD.
MONTALE, NJ 07048
201 882-3266
RICHARD@SYS-CON.COM

XMLEDGE:
conference & expo

JUNE 24-27
JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

SPONSORED BY:

IONA | END 2 ANYWHERE™

ADOS

bea

TogetherSoft

SilverStream

Actional

MERANT

Altworks

PolarLake
Enterprise through XML

ALTOVA | xml apps

MEDIA SPONSORS

Federal Computer Week

WebServices.org

XML TIMES.com

Java Skyline

GF Advisor

WebServicesHall

WIRE

XML.org

wrnx

ORACLE XML.org

SDTimes

JAVA SOURCEBOOK

wireless

XML

WebLogic

WebServices

WebSphere

Colofusion.com

OWNED AND PRODUCED BY

SYS-CON
MEDIA

SYS-CON
EVENTS



you the first available date, but you can schedule any available testing date and time. At this time, you'll have to pay for the test, which costs a little more than the Sun Java Programmer Certification Test, at \$200. After arranging payment by credit card (Visa, MasterCard, or American Express), you are ready to take the test.

On the day of the exam, be sure to arrive at the testing facility a few minutes early. Not only will this ensure that you're there on time, but also that you have the time needed to sign in. Early arrival may also calm any nerves you have before taking the test. Be sure to bring along photo ID and don't bother to bring any books, paper or pencils; they're not allowed. Everything you need will be provided.

After you take the test, you'll immediately be given a stamped score report with the results. It will indicate whether you pass or fail; a score of 72% is considered passing. Remember, since the test consists of 60 questions, you cannot miss more than 16! Be sure to hold on to the stamped score report at least until you receive your certification packet

because it's the only real evidence that you passed the test.

If you do pass, congratulations! You'll want to send a copy of your information to the BEA certification fulfillment center to receive a certification package that contains a framed copy of your certificate and a white BEA Certified Developer polo shirt (see Figure 1). Be sure to provide a letter with the following information:

- Name
- Address
- City, State, Zip, Country
- Phone number
- E-mail address
- Shirt size
- Name as you would like it to appear on the certificate
- Date you completed the test

Fax the letter and a copy of the BEA WebLogic Server 6.0 Developer Test score report to the fulfillment center at 408 570-8921. Be patient, though, the certification package takes three to six weeks to arrive after you submit your request. Be sure to confirm all of the above information at the BEA certification Web site: http://education.bea.com/education/certification_checklist.jsp.

If you fail the test, you're eligible to retake it as soon as possible. Unlike other vendors, BEA places no restrictions on retakes of the exam, so you can try as many times as needed – even on the very next day if you want to. Of course, you'll have to call Prometric, schedule the test, and pay \$200 for

every additional time that you have to take it. So, it's best to study very carefully for the exam so you can pass it on the first try.

Studying for the Exam

In my opinion, studying for this exam is a bit more difficult than studying for any other large vendor certification test. Many of the large vendors, such as Sun and Microsoft, have a lot of practice material that you can study to help prepare yourself for their certification tests. Unfortunately, the BEA WebLogic Server 6.0 test has very little study or practice material written especially for it. An exam study guide is available from the BEA certification home page (http://education.bea.com/education/study_guide.jsp), but it contains a series of Web links that may be too general to use as effective study material. A single sample test is available, but only for BEA Star Partners and not the general public; so for the most part, there is no practice exam available. The BEA Web site does provide a list of exam objectives. Based on that list, the test covers the following high-level topics:

- Designing and building enterprise components
- Designing and building Web components
- Demonstrating database connectivity
- Using transactions
- Configuring, packaging, and deploying enterprise applications
- Using vendor tools to monitor and manage the server
- Configuring the server

For the complete list of exam objectives, visit the BEA certification Web site at

http://education.bea.com/education/certification_test_objectives.jsp.

As you can see, these exam topics tend to be very general, making it difficult to decide where to start your exam preparation. Even the study guide Web links can be very general. For example, one requirement listed in the Study Guide (http://education.bea.com/education/study_guide.jsp) is "Design and build reusable enterprise components"; the two resources listed to study are www.componentsource.com/BuildComponents/WhitePapers/EJBWhitePaper.asp and <http://java.sun.com/>!

What It Covers

Having taken certification tests before, I knew how detailed some of the questions could be and I assumed that the WebLogic Server 6.0 test would be similar. As I was studying, I found it a little irritating not to have anything detailed to focus on. Therefore, I've gone through and simplified some of the study work for you! I've broken BEA's test objectives into functional areas



BEA Certified Developer shirt



that a typical developer can more easily associate with. Hopefully, it will make the test easier to study for overall.

WEB APPLICATIONS

Questions on Web applications represent a large portion of the test. The questions cover general topics on HTTP Servlets and JSP pages as well as using the HTTP Session and Servlet Context. Understanding error handling, deployment, and Web application security is critical to passing the test. Special attention should be given to understanding delegation models in Web applications (forward and include).

ENTERPRISE JAVABEANS

The next most important technology is Enterprise JavaBeans (EJBs). Understanding the difference between the types of EJBs (Stateless Session, Stateful Session, Entity, and Message Driven) is crucial. The test also covers: writing EJB clients, including JMS clients, using EJB exceptions, and using EJB utility classes, such as SessionContext, in EJB applications. Understanding EJB deployment and deployment descriptors is also critical. I've put JMS in the EJB section because JMS accounts for only a small percentage of the actual test. You should understand JMS well enough to know how a client sends a JMS message to a message-driven bean.

CLUSTERING

Surprisingly, a thorough understanding of the WebLogic Server 6.0 clustering features is important in passing the developer certification test. Understanding load balancing algorithms and how the high availability services function is also important.

DATABASE CONNECTIVITY

A good understanding of general database connectivity is essential. Some of the questions are on JDBC, Connection Pooling, and Datasources. Other topics, such as Multipools, should be understood as well.

TRANSACTIONS

A good understanding of the different types of transactions (container managed and bean managed) used in EJB applications is essential. Using the UserTransaction object is also a key topic.

JAVA NAMING AND DIRECTORY INTERFACE

JNDI accounts for a small percentage of the actual test, but understanding it is critical in answering many of the questions. An experienced J2EE developer should have no problems with this topic, but be sure to spend some time

covering the "java:comp" naming convention if you are not familiar with it.

Other Topics

Before you take the test, you should have a good knowledge of the WebLogic Server 6.0 Administration Console. In particular, understanding the server security model is vital, as is how to use the administration console to deploy EJBs and applications. Auxiliary topics, such as JMX and CORBA, will also be covered.

It's so important, it's worth mentioning twice. Be sure to go through the Server Administration Console and look at some of the different options you can set and monitor.


Conclusion

I'm sure this seems like a lot to study! You're probably thinking that I haven't really reduced the amount you have to study at all! But don't worry, in upcoming articles, I'll choose a topic (or

" Passing the BEA WebLogic Server 6.0 Certification Test is the key to becoming a BEA Certified Developer "

two) and provide more details. Explanations and sample questions (and answers) will be provided for all of the topics mentioned above. Some of the detailed explanations may seem too basic for some of you, but the sample questions will help even the most experienced Weblogic Server developer.

Finally, passing the BEA WebLogic Server 6.0 Certification Test is the key to becoming a BEA Certified Developer. Hopefully, I've provided you with enough information to start studying and sign up to take the Test. If you want more general information on the certification program, be sure to visit BEA's certification site at http://education.bea.com/education/certification_program.jsp. Upcoming articles will offer more detailed coverage of the high-level test topics . Using these tools, along with your hands-on experience with BEA WebLogic Server 6.0 and some real studying, you should be able to pass the test and become a BEA Certified Developer.

Good luck! 

This month I'm going to look at some of the things I found in those days when I studied the use of SQL with WebLogic 6.1.

The Benefits of SQLj

REPLACING JDBC CODE WITH SQL - MAKING LIFE EASIER

BY MIKA RINNE



AUTHOR BIO...

Mika Rinne is a senior consultant with BEA Systems Inc. He has been programming since he was 13 and in 1995 built an Internet and BEA TUXEDO-based online ticket-selling service, the first of its kind in Scandinavia.

CONTACT...

mika.rinne@bea.com

I got into the subject because I needed to go through a lot of existing JDBC code. As you may know, JDBC code is not the easiest to read, which made me start to think about embedded SQL, because of its better readability compared to JDBC.

In the past, I got used to writing applications on BEA's Tuxedo application server using the C language and embedded SQL, namely PRO*C, Oracle's name for their embedded SQL in C. I recalled that it worked nicely and had some very nice features like compilation time syntax checking and readability, features that JDBC is, obviously, lacking.

That got me studying SQLj, which is the name for embedded SQL in Java and which you can use, for example, with Oracle databases. I found lots of potential uses for writing SQL code instead of JDBC in Java, such as building Enterprise JavaBeans. In this article, when I talk about SQLj, I mean the implementation of SQLj from Oracle release 8.1.7.

Basics

First of all, embedded SQL means placing your SQL statements directly into your Java without using String objects (and without a bunch of other objects from the JDBC API), in contrast to JDBC. That gives you better readability for your code because the statements are pretty much like those you can issue from SQL*Plus or other SQL tools. For example:

```
#sql select ENAME, SAL from EMP order by ENAME;
```

After placing these commands in your code, you need to precompile the source code into a .java source, giving you the second important feature in SQLj, which is the compilation time syntax checking. In contrast to SQLj, in JDBC you need to compile and deploy your JDBC application, start the server, and run the application just to find out that the syntax of the SQL statement is incorrect. In SQLj, however, you don't necessarily have to carry out all that, since you'll get the error in precompilation time for syntax incorrectness. That said, I must add that the syntax checking could be much better – in most cases you find out about your SQL syntax errors during runtime.

There might also be a third reason for using SQLj: the performance. Some people say that it's better when compared to JDBC, but since I didn't do any tests and didn't measure it, I can't say. But according to the tests I did with EJBs, I can say the performance is certainly acceptable.

In order to write a SQLj-based Java application you first have to create a .sqlj source file like MyClass.sqlj, which contains the embedded SQL, and then compile it to a .java file, in this case MyClass.java, using the SQLj precompiler. Finally, the MyClass.java source file is compiled into a Java class file for execution, MyClass.class, accordingly.

The SQLj compiler is logically named 'sqlj' and is capable of not only precompiling the embedded SQL, but also of compiling the Java source into a Java class or classes. Alternatively, you can execute your own java compilation phase after the precompilation if you are explicitly setting the sqlj not to compile classes with the -compile option set to "false" (there's also a lot of other options as well). The procedure is as follows (as seen on the command line in Windows 2000):

```
\> set CLASSPATH=../lib/translator.zip;../lib/runtime.zip
\> sqlj MyClass.sqlj
```

or

```
\> set CLASSPATH=../lib/translator.zip;../lib/runtime.zip
\> sqlj -compile=false MyClass.sqlj
\> javac MyClass.java
```

This will create MyClass.class in both cases, with the exception that in the first example the "default" Java compiler is used and in the latter

the explicitly specified javac Java compiler from Sun's JDK is used. The latter also usually provides better output than the former in case of errors in the java source according to testing, especially when using Ant for building (see Building the EJB Using Ant later in this article).

Example One: A Local Class

The first example is the source code for the MyClass, which demonstrates the use of a SQL query we defined earlier and the use of the WebLogic OCI Driver for Oracle within SQLj. The implementation could be as shown in Listing 1.

When compiled and run from command line, the result would look like Listing 2.

Before running it you need to set the WebLogic classpaths, which you can do by running the config\examples\setexamplesenv.cmd from your WebLogic 6.1 installation directory.

If there had been errors in the SQL, the pre-compiler might have detected them, for example:

```
>sqlj -compile=false MyClass.sqlj
MyClass.sqlj:24.9-24.55: Error: SQL statement could not
be categorized.
Total 1 error.
```

Although, as I said earlier, this is not a typical case; when you use SQLj you will still usually find out about your syntactical errors during runtime.

Example Two: An EJB

While the first example is a static class and run locally, our second example demonstrates the use of SQLJ within an Enterprise JavaBean (EJB).

There are two major areas of difference between a local class and a class (or classes) run on the server side inside WebLogic, like an EJB:

- Multithreading and concurrency
- Pooled database connections

Although these may be quite obvious to most readers, let's go through them quickly.

First, EJBs are always executed within threads, unlike most local classes, which means that clients execute the same piece of code simultaneously (within the same JVM, i.e., within the WebLogic instance).

Second, since the same piece of code runs simultaneously, we need to have pooled database connections for maximum performance. Otherwise, we would either run out of available database connections (i.e., exceed the maximum number of connections from the database point of view) or, if there is only one connection (and synchronization is used), the performance would be very bad.

In SQLj we can do multithreading by first creating a so-called context and then using that context in our SQLj statements, for example:

```
#sql context MyContext;

#sql [myContext] myIterator = { select ENAME, SAL from
EMP order by ENAME };
```

Before you can use the context, it needs to be created from a pooled connection using a WebLogic OCI pool driver and a data source named "demoPool" of an EJB as follows:

```
InitialContext initCtx = new InitialContext();
DataSource ds = (javax.sql.DataSource)
initCtx.lookup("java:comp/env/jdbc/demoPool");
MyContext myContext = new
MyContext(Oracle.getConnection(ds.getConnection()));
initCtx.close();
```

If you aren't familiar with WebLogic data sources, take a look at the WebLogic EJB examples and documentation for more information.

The EJB implementation using the issues described above could be as shown in Listing 3. The query() method of this EJB returns a Vector of Emp records with ename and sal that we saw in the first example to the calling client.

Building the EJB Using Ant

The easiest way to compile and build the example MyBean EJB is to use Ant, which comes with WebLogic Server examples. I extended Ant's build.xml definition file a bit in order to execute the SQLj precompilation before the Java compilation (see Listing 4).

You must also include the step to the target name="all" in the build.xml, for example:

```
<target name="all" depends="clean,
init, sqlj, compile_ejb, jar_ejb,
ejbc,
compile_webapp,
compile_client"/>
```

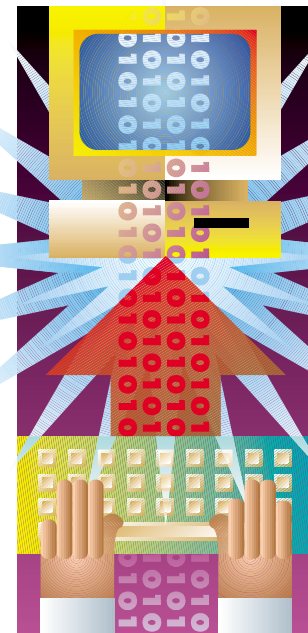
You should also set some properties for the task at the beginning of the build.xml:

```
<property name="libs" value="${source}/lib"/>
<property name="sqljlib1"
value="${libs}/translator.zip"/>
<property name="sqljlib2" value="${libs}/run-
time12.zip"/>
```

(These JAR files must also exist in the "libs" directory under your project directory.)

Running the EJB Client

After building the EJB with Ant and deploying it to WebLogic Server, you can run the EJB client



for testing. There's nothing special in the client to any other EJB client, since all SQLj code relies on the server side in the EJB.

However, I extended one of the EJB clients that come with the WebLogic examples a bit in order to make it multithreaded for optimal testing. I modified the client so it launches 50 client threads that will then call the query() method of the EJB. Then I started a couple of clients running simultaneously on different JVMs on my laptop, and had no problems with the EJB returning a response relatively quickly to each of the calling client threads.

Conclusion

The testing showed that during the execution

the number of pool connections grew to the maximum number of EJBs specified in the pool (set to 10 in the deployment descriptor), but after all threads had executed, the number of reserved pool connections dropped to zero. Thus we can say that the connection pooling worked just as it should with our example SQLj EJB.

A complete example of an SQLj EJB can be found on the **WLDJ** Web site at www.sys-con.com/weblogic/sourcec.cfm. The example can be also found on BEA's developer site <http://dev2dev.bea.com> under WebLogic Server examples.

More information about SQLj can also be found on Oracle's Web site with the SQLj compiler version 8.1.7 used in these examples.

Listing 1

```
import java.sql.*;
import java.util.*;
import oracle.sqlj.runtime.Oracle;
#sql iterator MyIterator (String, double) ;
public class MyClass
{
    public static void main(String[] args) {
        try {
            Properties props = new Properties();
            props.put("user", "scott");
            props.put("password", "tiger");
            props.put("server", "MJR");
            Driver ociDriver = (Driver)
            Class.forName("weblogic.jdbc.oci.Driver").newInstance();
            DriverManager.registerDriver(ociDriver);
            Connection conn = ociDriver.connect("jdbc:weblogic:oracle",
            props);
            Oracle.connect(conn);
            MyIterator myIterator = null;
            String ename = null;
            double sal = 0;

            #sql myIterator = { select ENAME, SAL from EMP order by ENAME };

            while (true) {
                #sql { FETCH :myIterator INTO :ename, :sal };
                if (myIterator.endFetch()) break;
                System.out.println(ename + " " + sal);
            }
            myIterator.close();
            Oracle.close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Listing 2

```
\>java -classpath %CLASSPATH%; MyClass
Starting Loading jDriver/Oracle .....
ADAMS 1100.0
ALLEN 1600.0
BLAKE 2850.0
CLARK 2450.0
FORD 3000.0
```

```
JAMES 950.0
JONES 2975.0
KING 5000.0
MARTIN 1250.0
MILLER 1300.0
SCOTT 3000.0
SMITH 800.0
TURNER 1500.0
WARD 1250.0
```

Listing 3

```
import java.sql.*;
import java.util.*;
import java.io.Serializable;
import javax.ejb.*;
import javax.naming.*;
import javax.sql.DataSource;
import oracle.sqlj.runtime.Oracle;

#sql context MyContext;
#sql iterator MyIterator (String, double) ;

public class MyBean implements SessionBean {
    private SessionContext ctx;
    public void ejbActivate() {}
    public void ejbCreate () throws CreateException {}
    public void ejbRemove() {}
    public void ejbPassivate() {}
    public void setSessionContext(SessionContext ctx) {
        this.ctx = ctx;
    }

    public Vector query() {
        MyContext myContext = null;
        try {
            myContext = init();
            MyIterator myIterator = null;
            Vector v = new Vector();
            String ename = null;
            double sal = 0;

            #sql [myContext] myIterator =
            { select ENAME, SAL from EMP order by ENAME };

            while (true) {
                #sql { FETCH :myIterator INTO :ename, :sal };
                if (myIterator.endFetch()) break;
```



```

        v.addElement(new Emp(ename, sal));
    }
    myIterator.close();
    return v;
} catch (SQLException sqle) {
    throw new EJBException(sqle);
}
finally
{
    cleanup(myContext);
}
}

private MyContext init()
throws SQLException
{
    InitialContext initCtx = null;
    try {
        initCtx = new InitialContext();
        DataSource ds = (javax.sql.DataSource)
        initCtx.lookup("java:comp/env/jdbc/demoPool");
        MyContext myContext = new
        MyContext(Oracle.getConnection(ds.getConnection()));
        return myContext;
    } catch(NamingException ne) {
        throw new EJBException(ne);
    } finally {
        try {
            if(initCtx != null) initCtx.close();
        } catch(NamingException ne) {

```

```

        throw new EJBException(ne);
    }
}
}

private void cleanup(MyContext myContext) {
    try {
        myContext.close();
        Oracle.close();
    } catch (Exception e) {
        throw new EJBException (e);
    }
}
}
}

```

Listing 4

```

<!-- Run the sqlj precompiler -->
<target name="sqlj">
    <echo message="Using ${sqljlib1}"/>
    <echo message="Using ${sqljlib2}"/>
    <exec dir="${source}" executable="sqlj">
        <arg line="MyBean.sqlj"/>
        <arg line="-status"/>
        <arg line="--compile=false"/>
        <arg line="-d=${build}"/>
        <env key="CLASSPATH"
        path="${java.class.path}:${sqljlib1}:${sqljlib2}"/>
    </exec>
</target>

```

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



BY
DAN MACKINNON

AUTHOR BIO...

Dan MacKinnon, a senior software engineer at WebGain Inc., was lead developer of the container-managed persistence integration for several releases of TopLink for WebLogic, and is currently working on EJB- and J2EE-based product initiatives. Dan holds a BSc and MSc in mathematics from Dalhousie University, and a BCS from Carleton University.

CONTACT...

dan.mackinnon@webgain.com

Exploring WebLogic JMX

PART 2

Creating WLS management applications

The Java Management Extensions (JMX) API provides a standard way of adding management capabilities to Java applications. BEA WebLogic 6.1 provides a full implementation of the JMX 1.0 specification, with all of its management features based on the JMX standard. As a result, the management capabilities in WLS are open and extensible, which makes it easy to build specialized management utilities for applications deployed on WLS.

In Part 1 of this series (*WLDJ*, Vol. 1, issue 4), we saw how JMX represents manageable resources as **Managed Beans** (MBeans), and provides a set of APIs for management applications to access those resources. In this article, we'll look at some of the details of how WebLogic Server 6.1 is instrumented using JMX, and how the JMX services offered by WebLogic can be used to monitor deployed applications. Using WLS JMX, we will gather runtime information from the application as it services client requests using the Java Pet Store example.

Before looking at the Pet Store example, though, we'll examine a few of the features of WebLogic JMX that simplify the creation of management applications.



Accessing MBeans in WebLogic Server

Management applications access the instrumented resources of a system to perform tasks such as monitoring or administration. These applications may take the form of graphical resource monitors, Web-based administration consoles, report-generating utilities, or other similar applications.

JMX allows the management interface of an application or resource to be exposed as a set of MBeans. These MBeans can then be accessed by management applications through the JMX *agent level*, whose main component is the MBeanServer.

WebLogic JMX is implemented in a way that simplifies the creation of management applications by making it easier to access MBeans and invoke methods on them. To gain access to MBeans, WebLogic JMX provides an **MBeanHome** – an augmentation of the standard MBeanServer that provides direct access to all the WebLogic Server predefined MBeans through a simplified API. The standard MBeanServer, by contrast, can provide only indirect access to the MBeans of the system. Access to MBeans is further simplified through WebLogic Server's use of JNDI and RMI in its JMX distribution level – providing direct access to the MBeans for both remote and local clients. Finally, all of the predefined WebLogic MBeans are implemented as *standard* MBeans (as opposed to dynamic and model MBeans, described in Part 1), which allows clients to make direct use of interfaces, instead of invoking MBean methods through reflection.

Together, these enhancements within the WebLogic Server JMX implementation simpli-

fy the creation of management applications that are specifically targeted for WebLogic Server. Applications that are meant to be portable to other application servers can still be written using the standard JMX APIs.

To observe the difference between the standard approach to accessing MBeans and the simplified approach using the WLS specific features, we'll look at two code samples, one that doesn't make use of these features and one that does. Listing 1 shows how to use standard JMX protocols to access MBeans in WebLogic Server. In particular, this example shows how to look at the "ServerRuntime" MBean, an MBean that is provided by WebLogic Server to supply overall information about the runtime system. After obtaining a reference to the MBeanServer, a JMX *ObjectName* instance is created using the full MBean name. The full name of this MBean is:

```
"petstore:Location=petstoreServer,Name=petstore
Server,Type=ServerRuntime"
```

Before the colon, we see the domain name; additional attribute/value pairs identify the MBean from others deployed on that domain. Once the *ObjectName* is created, it is passed to the MBeanServer, along with the name of the method to be invoked on the bean. The method "getWeblogicVersion" is called through the "invoke" method defined on the MBeanServer, which uses reflection to call the corresponding method on the *ServerRuntime* MBean – the client never directly accesses the MBean itself.

In contrast, Listing 2 shows how this process works using the simplified WLS interfaces. In this case, a reference to the MBeanHome is obtained. From there, we can get a direct reference to the *ServerRuntime* MBean using a much simpler lookup name than the full *ObjectName* of the bean (we only need to supply the "Name" and "Type" – the MBeanHome is already aware of the domain and server that it's deployed on). The *ServerRuntime* MBean can be sent messages just like any regular Java object; there's no need to use the reflective "invoke" protocol required by the MBeanServer.

There are several advantages to the WebLogic approach to looking up and using MBeans. The code is simpler – using direct invocation rather than reflection makes the code easier to write and understand. Moreover, using the interfaces directly means that type checking can occur – many coding errors will be caught at compilation, while the "reflective" style is more prone to error.

In addition to providing an augmented

and simplified API for accessing MBeans, WebLogic provides a large number of MBeans so that applications deployed on the server are "preinstrumented" and ready to be configured and monitored by management applications. Consequently, users of WebLogic JMX will generally find that they can create management applications based solely on the MBeans provided by WebLogic Server – only less frequently will users have to create their own MBeans to add additional instrumentation.

To get an appreciation of the amount of instrumentation that WLS provides with a deployed application, we can use MBeanHome to count the number of MBeans made available. Listing 3 shows how the MBeanHome can be queried for information about the MBeans available in the system. Running the code with the default "clean" domain (*mydomain*) and comparing this with the number of MBeans made available for the domain that has the Pet Store deployed in it shows that 577 MBeans are in the system for a clean WebLogic Server installation, but that 697 MBeans are in the system for the domain running the Pet Store. These 120 additional MBeans are beans used to configure the resources required by the Pet Store application (the JDBC connection pools, for example), and beans that can be used to monitor the state of the resources and components that make up the Pet Store.

Our main purpose is to show how these additional MBeans, created by WebLogic Server for deployed applications, can be used to carry out some simple runtime monitoring of the Pet Store. However, you may be wondering, "What are all these hundreds of MBeans in the default domain doing?"

WEBLOGIC SERVER MBEANS

Many of the MBeans that live within the server are there to provide an API for configuring the server itself. All of the configurable aspects of the WebLogic Server are represented as MBeans – from security, connection pools, and clustering, to the servers, and domains themselves. To get an idea of the scope of what can be configured, you only have to look at the WebLogic Administration Console. Virtually every element configurable through the console is also available through an MBean. Providing this layer means that all of the manual configuration that is normally done through the console can also be done programmatically, and that other utilities can provide the same functions as the console.

In addition to the configuration information represented as MBeans, a large amount



of transient runtime data is also available in MBean form. This includes information about open servlet sessions, active JDBC connections, JTS transactions, pooled EJBs, JMS messages, and so on.

To provide access to this configuration and transient runtime data, WebLogic Server provides three types of MBeans: Administration, Configuration, and Runtime.

ADMINISTRATION AND CONFIGURATION MBEANS

Administration and Configuration MBeans represent the configuration data found in the config.xml file and accessible through the Administration Console. Administration MBeans represent this data in its persistent form – any changes to these MBeans will be saved to the config.xml file. Configuration MBeans represent this data in its active form; changes made to these beans will affect the running system, but won't be saved to the config.xml file. The distinction between Administration and Configuration MBeans gives management applications the option of making either temporary changes to the configuration of the running system (through the Configuration MBeans), or permanent changes (through the Administration MBeans).

Administration and Configuration MBeans implement the same interfaces – they're differentiated based on their ObjectName and in how

The large numbers of MBeans in the “empty” domain can be attributed to the one Web application that is deployed to it – the Administration Console. The vast majority of the MBeans deployed in the “default” domain are actually MBeans associated with the Administration Console. Over 400 of the 577 MBeans that were counted in the default domain are associated with the console. These MBeans are runtime, administration, and configuration beans for the servlets and other components that make up the console, or additional MBeans created by the console to help it more easily administer the server.

WHY RUNTIME MONITORING?

Many of the MBeans provided by WebLogic Server are involved with tracking the evolving runtime state of the system and the application and resource components that reside in it. This runtime information can be very useful for application developers and for those administering already deployed systems.

Building runtime monitoring into a system is difficult – it often takes the clumsy form of “System.out.println” statements scattered throughout the code. Sometimes the monitoring can interfere with the running code, and in some cases what needs to be monitored is inaccessible – information that is hidden within a resource that the application is accessing.

The runtime monitoring provided by WebLogic JMX provides a better alternative to putting monitoring code directly into an application – and since the monitoring is built into the application server, it can track resource-related state that may be difficult to access from within the application code.

Runtime MBeans allow monitoring facilities to be built around an already existing and deployed application – without taking down the server or affecting the application in any way.

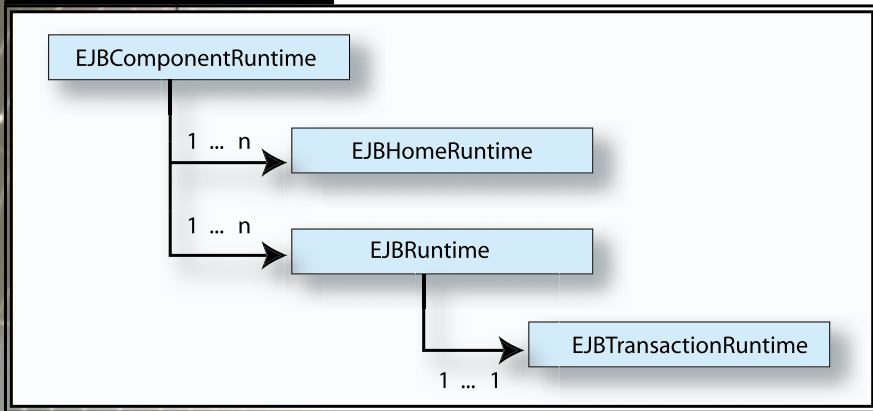
As an example, in the next section we will show you how to obtain some runtime statistics about the transactional behavior of some of the components within the Java Pet Store.

RUNTIME MONITORING OF THE JAVA PET STORE

The Java Pet Store application is the canonical J2EE example application. The Pet Store uses EJBs, JSPs, JMS, and other Enterprise Java features, and is built using an application architecture based on the J2EE blueprints described by Sun. It's a classic “online shopping” application based around the well-known “shopping-cart” paradigm for tracking purchases as users browse through an online merchandise inventory.

A version of the Java Pet Store is provided with the WebLogic 6.1 installation – a fully configured domain for the Pet Store is found in the “config” folder under the WebLogic home directory. All of the discussion and examples here use this Pet Store example, unmodified. The Pet Store included in the WebLogic 6.1 install is slightly modified from Version 1.1.2 of Sun's Java Pet Store.

FIGURE 1



Runtime MBeans for EJBs

they can be accessed. While Configuration MBeans are obtained from the MBeanHome as shown in Listing 2, Administration MBeans must be obtained from a distinct Administration MBeanHome. Each WebLogic Server instance has its own MBeanHome, while the Administration MBeanHome resides only on the administration server and is shared across a domain.

RUNTIME MBEANS

Runtime MBeans provide information about the state of deployed components (EJBs, servlets, etc.), resources (JDBC connection pools, sockets, etc.), and the server.

THE LARGEST INDEPENDENT

JAVA DEVELOPER CONFERENCE IN THE WORLD!

WIN A \$35,000 LUXURY CAR!



ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

JAVA IN JUNE ESPECIALLY IN NEW YORK

REGISTER ONLINE TODAY

FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT WWW.SYS-CON.COM

Focus on Java

Java, now mainstream, is the dominant back-end technology upon which next-generation technologies are evolving.

Hear from the leading minds in Java how this essential technology offers robust solutions to technology professionals and senior IT/IS strategy decision makers.

The Java Fall Rally on June 24, a Java-focused CEO Roundtable Panel, and comprehensive conference sessions will focus you on Java-only information-packed day!



A Sampling of Java-Focused Sessions

- JAVA 1.4: WHAT'S NEW?
- BUILDING TRULY PORTABLE J2EE APPLICATIONS
- JAVA TOOLS FOR EXTREME PROGRAMMING
- BUILDING ASYNCHRONOUS APPLICATIONS USING JAVA MESSAGING
- JET VS. J2EE
- J2EE: SETTING UP THE DEVELOPMENT ENVIRONMENT
- BUILDING WEB SERVICES WITH J2EE
- DETECTING, DIAGNOSING, AND OVERCOMING THE FIVE MOST COMMON J2EE APPLICATION PERFORMANCE OBSTACLES



ALAN WILLIAMSON
JAVA CHAIR • EDITOR-IN-CHIEF
AND DEVELOPER JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 J-TECHNOLOGY PROFESSIONALS

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- J-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...



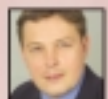
TYLER JEWELL
PRINCIPAL TECH
EVANGELIST • BEA



GREGG KIESSLING
CEO
SITRAMA



DAVID LITWACK
CEO
SILVERSTREAM



BARRY MORRIS
CEO
IONA



GREGG O'CONNOR
PRESIDENT
SONIC SOFTWARE



RICK ROSS
FOUNDER
JANLOBBY



JAMES DUNCAN
DAVIDSON
FATHER OF ANT

For Exhibit Information

CONTACT: MICHAEL PESICK
135 CHESTNUT RIDGE RD.
MONTVILLE, NJ 08055
201-982-3957
MICHAEL@SYS-CON.COM



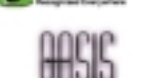
JUNE 24-27
JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

SPONSORED BY:



MEDIA SPONSORS





TRANSACTION MONITORING EXAMPLE

For our runtime-monitoring example using WebLogic JMX, our goal will be to find how many JTS transactions are committed, rolled back, or timed-out against the “Shopping Cart” component in a typical user interaction with the Pet Store. We’ll do this with a small Java class `CartTransactionTracker` that connects to the running server remotely.

In an EJB-based application, transactional overhead can be a significant performance issue, so this kind of monitoring is quite relevant. By extending this example to the other components in the system, it’s possible to detect the components that are the most “transaction intensive.”

There are actually several ways to get information about the transaction load from WebLogic’s JMX services. The method that we’ll use involves looking at several Runtime MBeans that are associated with deployed EJB components.

Information about the transactions committed, rolled back, or timed out against the shopping cart EJB can be obtained from the `EJBTransactionRuntime` MBean associated with it. Each EJB deployed in WebLogic has the `EJBTransactionRuntime` MBean that carries this information.

To look up the `EJBTransactionRuntime` for a particular bean, there are, again, several methods. Every MBean in the system can be looked up directly based on its full name. However, WebLogic Server provides a more intuitive system, by which MBeans can be accessed from other MBeans related to them.

A good entry point to the MBeans associated with a particular EJB is provided by the `EJBComponentRuntime`. The `EJBComponentRuntime` of an `ejb-jar` is one way to access the MBeans for all the EJBs deployed in that archive. From `EJBComponentRuntime`, we obtain a reference to the MBean associated with the particular bean – an `EJBRuntime`. The `EJBRuntime` maintains a reference to the `EJBTransactionRuntime`, which (as previously mentioned) tracks transaction information for a particular EJB. The relationship between these Runtime MBeans is shown in Figure 1.

In our example we want to find out about the transactions associated with the Shopping Cart EJB. This bean, whose EJB name is “TheCart”, consists of a bean-class, `ShoppingCartEJB`; a remote interface, `ShoppingCart`; and a home interface, `ShoppingCartHome`. It is deployed in the `ejb-jar` called “`shoppingcartEJB.jar`” within the “`petstore.ear`” enterprise archive.

To get to the `EJBRuntime` MBean associated with the Shopping Cart, we’ll look up the `EJBComponentRuntime` for the “`shoppingcart.jar`”. WebLogic’s naming convention for `EJBComponentRuntime` MBeans is straightforward – the `ejb-jar` name, followed by an underscore, followed by the EAR name; in this case, “`shoppingcartEJB_petstore`”. Listing 4 shows the code required to obtain the MBean for this `ejb-jar`.

The `shoppingcartEJB.jar` file contains two EJBs – the Shopping Cart entity bean, and the Catalog session bean. To find the Shopping Cart, we can ask the `EJBComponentRuntime` MBean for all of its beans, and select the bean with the matching EJB Name. Listing 5 looks at the `EJBRuntime` MBeans returned from the `EJBComponentRuntime` for the MBean with the `ejb-name` “TheCart”.

Finally, we can obtain information about the transactions from the `EJBTransactionRuntime` that is available from the `EJBRuntimeMBean`. Listing 6 briefly shows how this can be done.

Combining these steps – looking up the MBeans that represent the JAR, the individual EJB, and the transactions for the bean – yields a simple client that reports the transactional activity of the Shopping Cart EJB. Running this after single-user tests can provide some insight into bean behavior. Extending this across other EJBs in the system can reveal which components do the most transactional work, and can, in stress tests, detect points of failure.

Possible Extensions

What other information could we have obtained from the various EJB Runtime MBeans that we accessed in this example? Additional information includes caching, pooling, and locking data (all available on the bean type-specific subinterfaces of the `EJBRuntimeMBean`).

In this example, a simple remote client was used to log the results each time the client was run. Other implementation possibilities include:

- Having the program implement the `javax.management.NotificationListener` interface, and publish Transaction data as it becomes available. WebLogic JMX supports both remote listeners and listeners running inside the server.
- Using the various MBean monitors provided with JMX, we could set the logging options to record when the number of transactions rolled-back or timed-out reach certain threshold values.
- Creating a new MBean that ties together the relevant monitoring information for the Pet Store application. This `PetStoreRuntimeMBean` would access the other standard MBeans we discussed, but tie them together so that they are more specifically tailored for the particular application. The benefit of extending WebLogic Server Administration by adding new MBeans is that any generic JMX monitoring tool could then access this customized management interface. However, user-defined MBeans must be accessed through the `MBeanServer`, and not the `WLSMBeanHome`.

Other possibilities include extending the monitoring to cover servlet runtime behavior and JDBC connection pools.

Simplex Knowledge Company

www.skc.com/training



Conclusion

Many enterprise-critical applications have been deployed on the J2EE platform – a key requirement of such applications is the availability of flexible and useful management utilities.

WebLogic 6.1 and its JMX-based administration facilities provide several enhancements over the standard JMX API that make it easy to build simple management applications while still allowing users to create standard applications that can run against any JMX implementation. These can be used to monitor existing applications without affecting the application code, and can be particularly helpful in locating problem areas or generating data to help in performance tuning.

In addition to providing enhancements to the JMX distribution and access levels, WebLogic JMX provides a full instrumentation of the WebLogic Server. Administration, configuration, and runtime data can be easily accessed through simple applications running inside or outside the server.

Finally, WebLogic's management capabilities are extensible, which means that any custom interfaces built using JMX can be accessed by JMX-compliant management applications.

By taking advantage of the APIs and MBeans provided by WebLogic Server, developers can create applications that are easier to monitor and administer, and can also create sophisticated management tools for those applications.

References

- BEA Systems (2001). "Using WebLogic Server JMX Services." <http://edocs.bea.com/wls/docs/61/jmx/index.html>.
- Sun Microsystems. "J2EE Blue Prints" (includes Pet Store documentation). <http://java.sun.com/blueprints/enterprise/index.html>.
- Sun Microsystems (July 2000). "Java Management Extensions (JMXTM) Instrumentation and Agent Specification (JSR 3)." <http://jcp.org/jsr/detail/3.jsp>.
- Sun Microsystems. "Java Platform Enterprise Edition Specification, v1.3." <http://jcp.org/jsr/detail/3.jsp>.
- Sun Microsystems (June 1999). "Java Management Extensions White Paper: Dynamic Management for the Service Age"; Revision 01. <http://java.sun.com/products/JavaManagement/doc.html>

Listing 1: Use of standard JMX protocols to access MBeans

```
public void standardLookupExample() throws Exception{
    // Use the JMX standard MBeanServer
    MBeanServer mbServer = lookupServer();

    // The full MBean name must be used, and an ObjectName instance constructed
    String stringName =
        "petstore:Location=petstoreServer,Name=petstoreServer,Type=ServerRuntime"
```

```
ObjectName beanName = new ObjectName(stringName);

//The method is invoked reflectively using the MBeanServer's "invoke" method
String result = null;
result = (String) mbServer.invoke(beanName, "getWeblogicVersion", null, null);
log("WebLogic Server Version is " + result);
}
```

Listing 2: Use of WLS MBeanHome to access MBeans

```
public void directLookupExample() throws Exception{
```

RECEIVE \$150
DISCOUNT OFF FULL CONFERENCE
WEB SERVICES EDGE REGISTRATION



**Learn How to Develop
SOAP Web Services NOW!**
at a One-Day Tutorial... Coming to a City Near You!


```
// Use the WLS MBeanHome interface
MBeanHome mbHome = lookupHome();

// Obtain a direct reference to the desired MBean, based on its short
name, and type
ServerRuntimeMBean serverRuntime = (ServerRuntimeMBean)
    mbHome.getRuntimeMBean("petstoreServer", "ServerRuntime");

// Invoke the method directly on the MBean
String result = serverRuntime.getWebLogicVersion();
log("WebLogic Server Version is " + result);
}
```

Listing 3: Counting available MBeans

```
public void countBeans() throws Exception{
    //Obtain the MBeanHome from the server
    MBeanHome home = lookupHome();
    //Use the MBeanHome API to count the MBeans deployed
    Set beans = home.getAllMBeans();
    log("The domain is: " + home.getDomainName());
    log("The number of MBeans is: " + beans.size());
}
```

Listing 4: Obtaining the EJBComponentRuntime MBean

```
public EJBComponentRuntimeMBean getEJBComponentRuntime()
throws Exception{
    String ejbjarName = "shoppingcartEjb_petstore";
    EJBComponentRuntimeMBean mbean = null;
    mbean = (EJBComponentRuntimeMBean) lookupHome()
        .getRuntimeMBean(ejbjarName, "EJBComponentRuntime");
    return mbean;
}
```

Listing 5: Obtaining the EJBRuntimeMBean

```
public EJBRuntimeMBean getEJBRuntimeMBean(
    EJBComponentRuntimeMBean componentMBean){

    EJBRuntimeMBean[] beans = componentMBean.getEJBRuntimes();
    EJBRuntimeMBean selectedBean = null;
    for(int i = 0; i < beans.length; i++){
        EJBRuntimeMBean ejbMBean = beans[i];
        if(ejbMBean.getEJBName().equalsIgnoreCase("TheCart")){
            selectedBean = ejbMBean;
            break;
        }
    }
    return selectedBean;
}
```

Listing 6: Obtaining and using the EJBTransactionRuntime

```
public void displayTransactions(EJBRuntimeMBean ejbMBean){

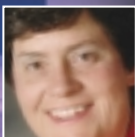
    log("Bean has JNDI name: " + ejbMBean.getName());
    log("Bean has EJB name " + ejbMBean.getEJBName());
    JTransactionRuntimeMBean txrt =
        ejbMBean.getTransactionRuntime();

    log("COMMITTED: " + txrt.getTransactionsCommittedTotalCount());
    log("ROLLED BACK: " +
        txrt.getTransactionsRolledBackTotalCount());
    log("TIMED OUT: " + txrt.getTransactionsTimedOutTotalCount());
}
```

Jump-start your Web Services knowledge. Get ready for Web Services Edge East and West!

AIMED AT THE JAVA DEVELOPER COMMUNITY AND DESIGNED TO EQUIP ATTENDEES WITH ALL THE TOOLS AND INFORMATION TO BEGIN IMMEDIATELY CREATING, DEPLOYING, AND USING WEB SERVICES.

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASE TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI, AND XML, AND MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS, AND REMOTE REFERENCES.



PRESENTERS...

Anne Thomas Manes, Systinet CTO, is a widely recognized industry expert who has published extensively on Web Services and service-based computing. She is a participant on standards development efforts at JCP, W3C, and UDDI, and was recently listed among the Power 100 IT Leaders by Enterprise Systems, which praised her "uncanny ability to apply technology to create new solutions."

Zdenek Svoboda is a Lead Architect for Systinet's WASP Web Services platform and has worked for various companies designing and developing Java and XML-based products.

EXCLUSIVELY SPONSORED BY



BOSTON, MA (Boston Marriott Newton) **SOLD OUT!** **JANUARY 29**
WASHINGTON, DC (Tysons Corner Marriott) **SOLD OUT!** **FEBRUARY 26**
NEW YORK, NY (Doubletree Guest Suites)..... **NEW DATE!**.....**APRIL 19**
SAN FRANCISCO, CA (Marriott San Francisco)**MAY 13**

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE \$495 REGISTRATION FEE.

Register at www.sys-con.com or Call 201 802-3069



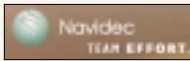
News & Developments

Mongoose, Insevo, Navidec to Deliver Application Integration

(Houston) – Mongoose Technology, a leading provider of enterprise portal solutions; Insevo, the market leader in J2EE Connector Architecture-compliant application integration software; and Navidec, a leading systems integrator specializing in portal implementation, announced that they are integrating their products and service offerings to create an



end-to-end solution for delivering scalable and flexible enterprise



portals with seamless integration into leading ERP and CRM applications. Using PortalStudio with Insevo connectors and Navidec portlet and integration expertise, customers can achieve significant savings over traditional portal and EAI offerings and enable much faster time-to-market and a rapid return on investment.

PortalStudio with the Insevo Connectors and Navidec Portlets and services will be released as part of PortalStudio Enterprise Edition Version 3.0. www.mongooasetech.com, www.insevo.com, www.navidec.com

BEA WebLogic Tops Benchmark Results for Performance and Value (San Francisco) – BEA Systems has further cemented its industry leadership with new world-record ECperf benchmark results for both performance and price/performance with BEA WebLogic Server 7.0, the latest version of its application server. The results show that BEA provides the most scalable solution and the best value,

from enterprise to departmental applications.

ECperf benchmark tests are an industry standard for measuring the performance and price/performance ratio of J2EE servers. The tests simulate actual business conditions to help organizations understand how any combination of application server, hardware, and database would perform and determine the associated costs in demanding application environments. Tests were conducted using BEA WebLogic Server 7.0 to evaluate the ability of the BEA platform to provide unbeatable power and scalability for heavy-duty enterprise applications, as well as ultra-low price/performance for applications running in departmental and workgroup settings. www.bea.com



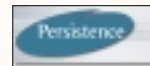
AdventNet Announces Immediate Availability of Middleware Manager 3.0 WebLogic Edition (Pleasanton, CA) – AdventNet, a market leader in standards-based Java and J2EE management solutions, has announced the immediate availability of



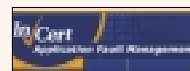
standards-based (JMX, Java SNMP, etc.) AdventNet Middleware Manager 3.0 WebLogic Edition. The product is available for full download at www.adventnet.com

AdventNet Middleware Manager WebLogic Edition is a real time, standards-based, best-of-breed product, offering a scalable, feature-rich and cost-effective solution to manage all components of an application ecosystem developed with BEA WebLogic Server. www.adventnet.com

Persistence Software Announces Real-Time Enterprise Initiative (San Francisco) – Persistence Software has announced its Real-Time Enterprise Initiative to help Global 2000 companies break the barriers of real-time visibility for vital business operations. With a new suite of solutions built on Persistence's patented distributed dynamic caching (DDC) technology, EdgeXtend for WebLogic Beta 1.0 and EdgeXtend DirectAlert will enable enterprises to create an effective distributed computing environment thereby empowering decision makers at the edge of the enterprise to have an up-to-date view into the state of business by eliminating critical data-access challenges. Such a capability will lead to the enterprise realizing enhanced productivity, better operations, and optimized business processes. www.persistence.com



InCert Offers Halo for J2EE (San Francisco) – InCert Software has announced the general availability of Halo for the J2EE platform. The latest version of Halo provides advanced application fault management capabilities to companies developing, managing and deploying enterprise-level J2EE applications. Halo for J2EE has been specifically engineered to monitor, pinpoint, report on and provide a source code-level root cause diagnosis of software faults in EJBs, servlets, JSPs and any other Java code.



Halo helps to ensure that deployed code is reliable and quickly repairable should problems occur. Since Halo is an "always on" technology, all the information needed to solve a problem is captured on the first fault, eliminating the need for problem replication and bug

reports. The J2EE version includes a trace file browser, special documentation, and best practices to optimize the product for J2EE environments. www.incert.com

PointBase Server to be Bundled with WebLogic Server 7.0

(Mountainview, CA) – PointBase, a leader in Java database technology for managing and synchronizing enterprise data among servers and mobile and pervasive comput-



ing devices has formed a strategic alliance with BEA Systems to accelerate the development of Java 2 Enterprise Edition (J2EE) applications.

PointBase will bundle its 100% pure Java RDBMS, PointBase Server, with WebLogic Server 7.0 to provide a reliable and scalable database for e-commerce, application servers, Web servers, and other Internet-based applications.

The PointBase suite of products is a set of horizontal technologies providing SQL database and data synchronization capabilities to Java applications running on any Java-enabled platform. PointBase Server is the only RDBMS certified through Sun Microsystems, third-party JDBC technology certification program. Customers who prefer a stand-alone database receive a full-function evaluation license for 90 days. www.pointbase.com



Thought, Inc.

www.thoughtinc.com

Sitraka

www.sitraka.com/performasure/wldj