

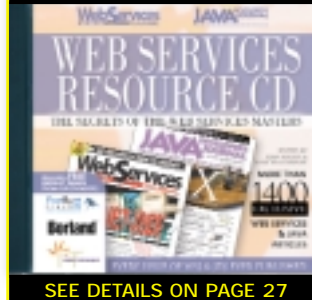
# BEA WebLogic

DEVELOPER'S JOURNAL

OCTOBER 2002 - Volume:1 Issue:10

weblogicdevelopersjournal.com

NOW SHIPPING



SEE DETAILS ON PAGE 27

FROM THE EDITOR  
Managing Complexity  
of J2EE  
by Jason Westra  
page 5

SAM'S SOAPBOX  
Using JMX  
by Sam Pullara  
page 28

WLS CERTIFICATION  
A Final Review  
by Dave Cooke  
page 30

WLDJ EDITORIAL  
Completing the  
J2EE Enterprise  
Nervous System  
by Sihyung Park &  
Oliver Schmelzle  
page 32

NEWS &  
DEVELOPMENTS  
page 50



TRANSACTION MANAGEMENT: **WebLogic Server System Administrator** Here's where the excitement begins     14  
Peter Holditch


INTEGRATION: **WebLogic Server on the Mainframe** PART 4 18  
Strategies for your biggest challenge Tad Stephens & Eric Gudgion

EJB QL: **The Power of EJB QL Subqueries in WebLogic Server 7.0** 22  
Satisfying search criteria   Thorick Chow

PORTAL: **Understanding the Advisor Framework** 34  
Demystifying the BEA WebLogic Portal Framework Dwight Mamanteo

SECURITY: **WebLogic Web Services Security** A look under the hood  38  
Anbarasu Krishnaswamy

COMMENTARY: **The Evolution Continues** 42  
Monitoring Web applications moves into transactions and Web services  Frank Moreno

PERFORMANCE: **Session Persistence Performance in BEA WebLogic Server 7.0** Finding the solution that best meets your needs  44  
Saurabh Dixit & Srikant Subramaniam

**Borland**  
[www.borland.com](http://www.borland.com)

**PANACYA**  
[www.panacya.com](http://www.panacya.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 EDITORS**  
JAMIE MATUSOW, JEAN CASSIDY

**ASSISTANT EDITOR**  
JENNIFER STILLEY

**WRITERS IN THIS ISSUE**  
THORICK CHOW, DAVE COOKE, SAURABH DIXIT,  
ERIC GUDGION, PETER HOLDITCH, JOE KROZAK,  
ANBARASU KRISHNASWAMY, DWIGHT MAMANTEO,  
FRANK MORENO, SIHYUNG PARK, SAM PULLARA,  
OLIVER SCHMELZLE, TAD STEPHENS,  
SRIKANT SUBRAMANIAM, JASON WESTRA

**SUBSCRIPTIONS**  
For subscriptions and requests for bulk orders,  
please send your letters to Subscription Department.  
SUBSCRIPTION HOTLINE:  
888-303-5282

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

**COO/CFO**  
MARK HARABEDIAN

**VP, BUSINESS DEVELOPMENT**  
GRISHA DAVIDA

**SENIOR VP, SALES & MARKETING**  
CARMEN GONZALEZ

**VP, PRODUCTION & DESIGN**  
JIM MORGAN

**ART DIRECTOR**  
ALEX BOTERO

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

**ASSISTANT ART DIRECTOR**  
TAMI BEATTY

**VP, SALES & MARKETING**  
MILES SILVERMAN

**ADVERTISING SALES DIRECTOR**  
ROBYN FORMA

**ADVERTISING ACCOUNT MANAGER**  
MEGAN RING-MUSSA

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

**VP, SYS-CON EVENTS**  
CATHY WALTERS

**CONFERENCE MANAGER**  
MICHAEL LYNCH

**REGIONAL SALES MANAGERS**  
MICHAEL PESICK • RICHARD ANDERSON

**ASSISTANT CONTROLLER**  
JUDITH CALNAN

**ACCOUNTS PAYABLE**  
JOAN LAROSE

**ACCOUNTS RECEIVABLE**  
KERRI VON ACHEN

**ACCOUNTING CLERK**  
BETTY WHITE

**WEBMASTER**  
ROBERT DIAMOND

**WEB DESIGNERS**  
STEPHEN KILMURRAY • CHRISTOPHER CROCE  
CATALIN STANCESCU

**ONLINE EDITOR**  
LIN GOETZ

**CUSTOMER SERVICE MANAGER**  
ANTHONY D. SPITZER

**CUSTOMER SERVICE REPRESENTATIVE**  
MARGIE DOWNS

**JDJ STORE MANAGER**  
RACHEL MCGOURAN

**EDITORIAL OFFICES**  
SYS-CON Publications, Inc.  
135 Chestnut Ridge Road, Montvale, NJ 07645  
Telephone: 201 802-3000 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

**SYS-CON MEDIA**

# Managing Complexity of J2EE



BY JASON WESTRA  
EDITOR-IN-CHIEF

There's no question about it – J2EE applications are tough, burly pieces of software. Often they require numerous servers, communicate over various protocols, and run on software from various vendors.

Let's examine a simple J2EE application in which everything, including the database, runs on one machine. In this case, the Web server and application server are a single instance of WebLogic, and the database is the one bundled with your version of WebLogic. Sounds pretty easy to manage, right? You just put your applications in the /applications directory and WebLogic deploys them for you. Database connectivity to the bundled database came out-of-the-box. Could it get any easier? Actually, it's just the opposite. It could get a lot *harder*.

To get closer to a production environment, let's add security in the form of SSL and a firewall. We just added two more vendors (certificate authority and firewall vendor), another machine, and another Web server install. Our application is looking more complex, yet still manageable. However, we really need a production database on its own server, so let's install a database server and configure WebLogic to point to it. If you're counting, that's another box, another vendor, another install, and a big headache trying to configure the connection settings. I'm not even done throwing hardware, load balancers, WebLogic clustering, e-mail servers, content management software, and legacy systems into the mix! Where am I going with this? My point is simply that J2EE applications are complex. They're complex to design, complex to build, and complex to manage.

What can you do to manage the complexity inherent in J2EE applications? You can focus on a number of areas to bring it to a manageable level.

First, utilize the features of your application server to their fullest potential. The BEA

WebLogic Server platform is designed to make the tasks of developers and administrators a breeze. For starters, take a look at WebLogic Workshop and the WebLogic Server console. Workshop's approach to automated deployment of Web services and easy-to-use wrappers around J2EE components like JMS and EJB makes development of complex applications easier than ever. The WebLogic Server console offers administrators an easy way to configure and monitor WebLogic servers and their applications, even across clusters.

Another option is to utilize third-party software specially designed for managing Web applications. There are numerous products, such as NPULSE ([www.npulse.com](http://www.npulse.com)) AppAssure, that integrate nicely with WebLogic's JMX (Java Management Extensions) administration system. These products offer root-cause analysis to help you determine exactly where an error in your application has occurred, even across multiple servers in an *n*-tiered scenario. Also, they generally offer lightweight monitoring that can run alongside your production application without hindering it with bandwidth consumption, or raising your storage requirements for statistics.

A common problem encountered in large production environments is lack of clarity on the resources and machines in the data center – their purpose, their availability, and even what versions of software and applications are currently running on them. You can solve these problems and many more by utilizing a J2EE data center management platform that automates time-consuming tasks for developers and system administrators. These tasks might be tracking software versions, or configuring a Web server tier for security and virtual hosting.

*continued on page 33*

## 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 Developer's Journal* for two years, where he shared his WebLogic experiences with readers.

CONTACT: [jason@sys-con.com](mailto:jason@sys-con.com)



# BUILDING BETTER BRIDGES

## ADVANCED J2EE/PEOPLESOFT INTEGRATION USING JMS

BY JOSEPH K. KROZAK

**AUTHOR BIO...**

Joseph K. Krozak is vice president of Technology Development for Krozak Information Technologies, Inc. (www.krozak.com), a supplier of advanced software solutions to Fortune 500 and midmarket companies. He has over 15 years of experience in the information technology industry, including 10 years of extensive experience building large, distributed object- and component-oriented systems.

**CONTACT...**

jkrozak@krozak.com

**E**nterprise Application Integration (EAI) is a very popular topic these days. Businesses, as well as government entities at the local, state, and federal level, are all struggling to integrate their critical yet disparate information systems.

Some of the most challenging integration efforts involve integrating enterprise resource planning (ERP) or customer relationship management (CRM) systems with new and existing custom applications. These enterprise software applications often have proprietary architectures and complex APIs, and typically use proprietary programming languages unfamiliar to the mainstream developer community. However, some

enterprise software vendors are attempting to make the integration effort more approachable. Using widely accepted standards such as XML and Web services, these vendors are attempting to build better bridges between their respective products and the world around them.

PeopleSoft, Inc., is one of the dominant vendors in the enterprise software market. The PeopleSoft product family spans such critical

business functions as human resource management, financial management, supply chain management, and customer relationship management. These products, along with competitive offerings from vendors such as Siebel, SAP, and Oracle, often manage the most important business logic and data in the enterprise. As such, they represent important assets to be leveraged whenever possible and appropriate.

This article is the first in a two-part series describing integration methods and illustrating the use of integration technology to enable mutual reusability between PeopleSoft and J2EE applications. This article will describe how the Java Messaging Service (JMS) and PeopleSoft's Integration Broker technology can be used to integrate PeopleSoft and J2EE applications seamlessly. The second article will describe how to accomplish the same goal using Web services technology.

### PeopleSoft Architecture and Integration Technology

Starting with version 8, PeopleSoft products leveraged the PeopleSoft Internet Architecture (PIA), allowing the full functionality of these applications to be accessed from within a Web browser. By eliminating the need for client software, PIA's comprehensive Internet accessibility enables customers to realize substantially lower deployment costs than were previously possible under traditional client/server architectures. In addition to these benefits, PIA transitions the PeopleSoft product suite to a standard, J2EE-based operating platform built on BEA technology. PeopleSoft 8.4, the latest version (and the focus of this article), uses WebLogic Server 6.1 and Tuxedo for its Web and application servers, respectively.

As illustrated in Figure 1, PeopleSoft's Web tier components are deployed as J2EE Web applications under WLS 6.1, while its back-end components fall under Tuxedo control and are accessible via Tuxedo's Java-based Jolt API. These servers can be colocated or distributed, allowing additional flexibility in establishing environments that are optimized for scalability and redundancy. PeopleSoft Web applications handle a number of critical functions, from serving standard application pages and managing the distribution of batch-generated report output, to retrieving and aggregating user-requested content through the PeopleSoft Portal product. One Web application, the PeopleSoft Integration Gateway, plays a key role in integrating PeopleSoft products with each other and with third-party systems, including J2EE- and Microsoft .NET-based applications.

The Integration Gateway is a component of PeopleSoft's Integration Broker (IB) technology, introduced in PeopleSoft 8.4 (see Figure 2). The Integration Broker is an XML-based messaging hub that supports both synchronous and asynchronous messaging between PeopleSoft and external applications. IB is composed of several components and services spanning the Web and application server environments. The Integration Gateway resides on the

Web server, providing inbound and outbound pathways for PeopleSoft applications. Other components, such as the internal publication and subscription agents that actually move information across the application server boundary, compose the Integration Engine, and reside on the application server under Tuxedo control. Ultimately, one should be familiar with both environments to successfully develop and administer an IB integration solution. Due to the constraints of this article, I'll focus on the Web tier and the Integration Gateway, where the bulk of the developer-relevant configuration tasks lie.

The gateway provides the interface to the broker, managing the actual receipt and transmission of messages between integrated systems. The gateway consists of several listening and target connectors that support inbound and outbound communications between external systems and the integration engine. The connectors support a range of communication protocols, including HTTP, JMS, POP3, SMTP, and FTP. Additional PeopleSoft and PeopleSoft 8.1 connectors enable gateways to communicate with other PeopleSoft 8.4 and older 8.1 systems, respectively. Some of these protocols are supported for both inbound and outbound communications, while others are supported in only one direction (see Figure 2).

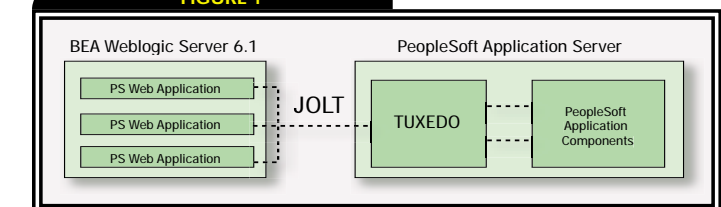
The Gateway Manager processes every message that enters through the Integration Gateway, invoking gateway services such as XML parsing, message validation, and logging. The Gateway Manager uses the content of the message object to determine the intended message recipient, and then uses the appropriate target connector to transmit the message. After receiving a response from the target connector, it forwards the response to the calling listening connector. During all of this processing, the contents of incoming messages are logged to a disk file, along with any errors encountered during processing.

A number of different physical architectures are possible, depending on the nature of the desired integration effort. For example, a simple integration, behind a company's firewall, may require a single local Integration Gateway. More elaborate integrations across firewall boundaries, LANs, and WANs could very well involve multiple remote Integration Gateway instances and several different listening and target connectors. As such, real-world PeopleSoft integration can get very complex, very quickly. This article presents some of the basics but will not delve into all of the possible integration scenarios, optimizations, and consequences.

### Integration Broker Definitions and Configurations

Every IB integration requires some degree of definition. The integration may involve several different systems communicating via multiple protocols and varying semantics. Hence, a number of conceptual definitions must be created and configured to appropriately model and implement the desired integration. The major IB definitions include gateways, nodes, transactions, relationships, messages, and message channels.

FIGURE 1



PeopleSoft Internet Architecture

- **Gateway:** Gateways handle incoming and outgoing messages through the broker. Each gateway has an address, which is the URL for that gateway's PeopleSoft listening connector, `http://<gatewayservername>/PSIGW/PeopleSoftListeningConnector`. Gateways use their PeopleSoft connectors to communicate with Integration Engines and other (remote) gateways. Additional connectors can be used to exchange messages with external systems using a wide range of protocols. Gateways can be remote or local. However, only one local gateway may be defined for any given PeopleSoft instance.

- **Node:** Nodes represent connection points in IB integrations, for example, other PeopleSoft applications or external systems. Nodes can be remote or local. However, for IB, all nodes are considered remote, except for the Default Local node representing the PeopleSoft instance being configured. A node's definition can reference a target connector, indicating the protocol (ex: JMS, HTTP, SMTP) the broker must use in order to reach that node.

- **Transaction:** Transactions are the most basic unit of work in an integration. They are associated with nodes, and designate which messages the node receives (inbound) and transmits outbound), and what semantics (i.e., synchronous, asynchronous) are used during the process. Every integration requires at least one transaction at each IB node either to send or to receive a message at the node.

- **Relationship:** Relationships reconcile transactional differences between the source and destination nodes of a given integration. For example, a given integration may define an inbound, asynchronous (inAsync) transaction of Message X at Node A, and an outbound, synchronous (outSync) transaction of Message Y, at Node B. In addition, Message Y may be a transformed, translated, or filtered version of Message X. Since, in this example, the semantics change from inAsync to outSync and the message formats differ, a relationship must be defined in order to resolve the differences and support the integration.

- **Message:** Messages represent the application data to be sent and received through the Integration Broker. Messages can be rowset-based or nonrowset-based. The former conveys hierarchical database information, based on native PeopleSoft structures called records. The latter can be used to convey practically any other message content or structure, including XML and Simple Object Access Protocol (SOAP) messages.

- **Message Channel:** Message channels are logical groupings of messages. Each message must belong to one and only one message channel. Message channel definitions are essential for sequenced processing, improved throughput, and message archival.

All of the IB definitions must be defined and configured using PeopleSoft's powerful PeopleTools Development Environment. Portions of PeopleTools are accessible via the PeopleSoft Internet screens (dubbed "PIA screens"), while other portions can only be accessed via the separate Application Designer client program. Gateway, node, transaction, and relationship definitions must be defined and configured via the PIA screens, while messages, message channels, and any custom PeopleSoft programs (written in a language called PeopleCode) must be created and maintained via the Application Designer.

### An Example: Integration via JMS

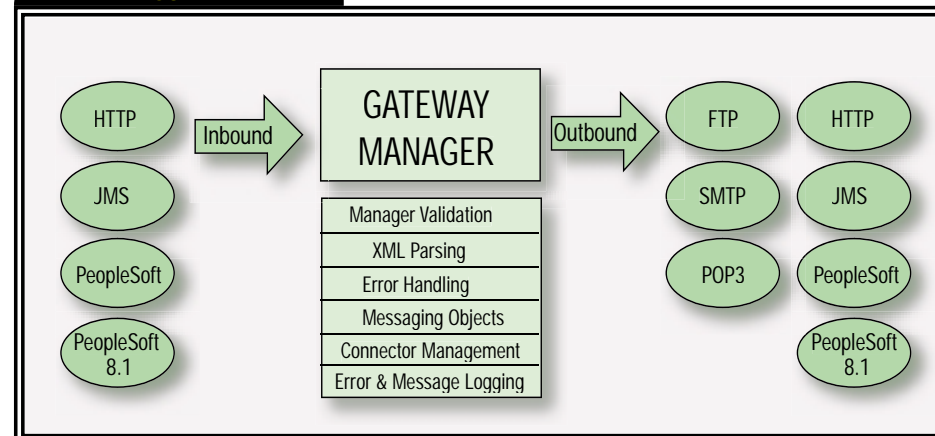
The following example will reinforce the concepts presented above. In this example, an integration scenario between WebLogic Server 7.0 hosted components and a single local PeopleSoft 8.4 instance using JMS will be explored.

One of the first steps in implementing an IB integration is to define the messages that will be exchanged between the participating systems. Two messages will be defined: one for messages emanating from WLS 7.0 to PeopleSoft, and another for messages traveling back. For the sake of simplicity, this example will use nonrowset messages with XML content.

Since any given message must belong to an existing message channel, the channel must be created first. After the message channel has been created, its configuration properties can be browsed. Figure 3 is an example of a message channel configuration dialog displayed in the Application Designer.

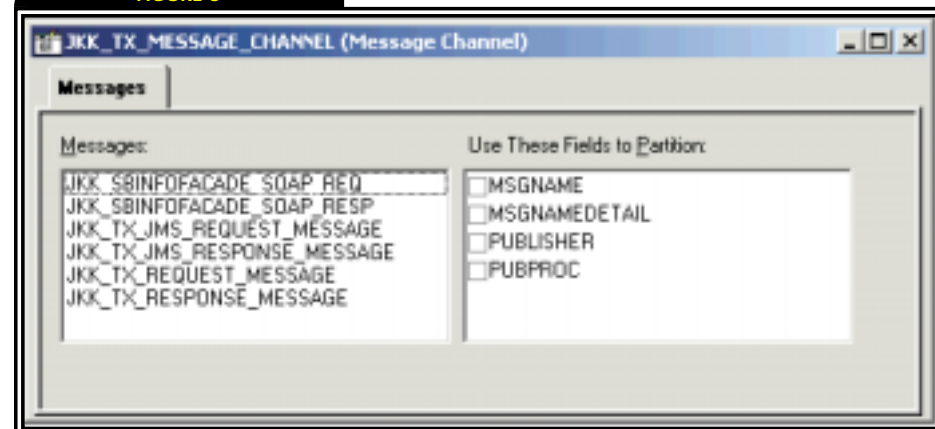
The JKK\_TX\_MESSAGE\_CHANNEL channel dialog displays the list of messages associated with the channel. In this example, the JKK\_TX\_REQUEST\_MESSAGE and

FIGURE 2



Integration Gateway architecture

FIGURE 3



Sample Message Channel and associated messages

# Dirig Software

[www.dirig.com/illusion/weblogic](http://www.dirig.com/illusion/weblogic)



JKK\_TX\_RESPONSE\_MESSAGE definitions represent the application data messages exchanged between WLS 7.0 and PeopleSoft. Typically, structured rowset messages will be defined in the Application Designer, along with a record-based content schema. However, for this simplified

example, the message definitions are simply monikers acting as proxies for the unstructured content they represent.

In this example, synchronous messaging will be used for both inbound and outbound traffic. Once messages arrive in PeopleSoft, they will be written to a disk file

before an empty response message is created and then returned to the WLS-hosted caller. After this, the original message will be routed back to WLS (over JMS) via a feedback loop.

In order to accomplish this, you must first write a PeopleCode request handler and attach it to the JKK\_TX\_REQUEST\_MESSAGE definition in the Application Designer. As in other development environments, such as Visual Basic, PeopleCode message objects can respond to various events. A message's OnRequest event can specify a custom PeopleCode routine to handle the synchronous arrival of the message. The PeopleCode routine can perform the appropriate actions before creating and returning a status message.

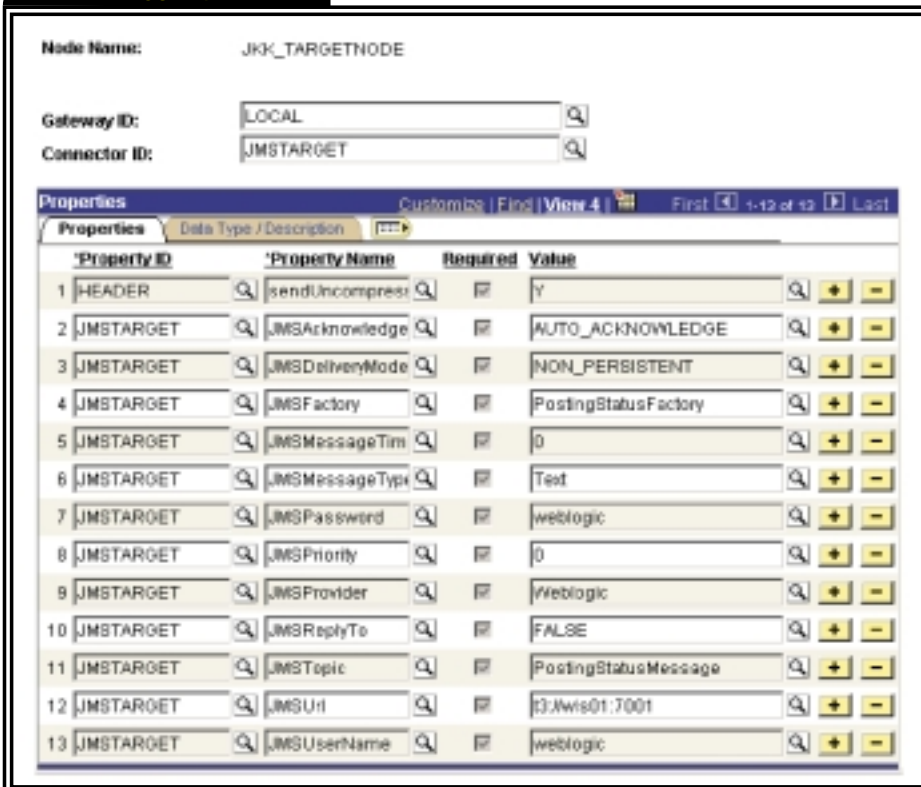
The PeopleCode request handler for the inbound JKK\_TX\_REQUEST\_MESSAGE is shown in Listing 1 (listings for this article can be found at [www.sys-con.com/weblogic/source.cfm](http://www.sys-con.com/weblogic/source.cfm)). As illustrated, the first programmatic instruction uses the special IB function, getMessageXmlDoc(), to retrieve the incoming XML message. After this, the message contents are converted into an XML string before being appended to a file named IncomingRequests.log. Finally, CreateMessage and ReturnToServer are special functions used to generate and then return an empty status message through IB back to the caller.

The next major step is to define and configure the necessary integration nodes, transactions, and relationships using the PeopleTools PIA screens.

Two nodes, JKK\_SOURCENODE and JKK\_TARGETNODE, will be created to represent the WLS instance participating in the integration. This instance will be the ultimate receiver of the messages it originally transmitted. The JKK\_SOURCENODE definition is fairly straightforward. However, the JKK\_TARGETNODE node definition is more involved because it must ultimately transmit messages to an external destination (the WLS instance) via JMS. Figure 4 provides a look at the target connector information attached to JKK\_TARGETNODE. Although there are several properties associated with the connector, we will focus on only the most important ones: JMSUserName, JMSPassword, JMSUrl, JMSTopic, and JMSFactory.

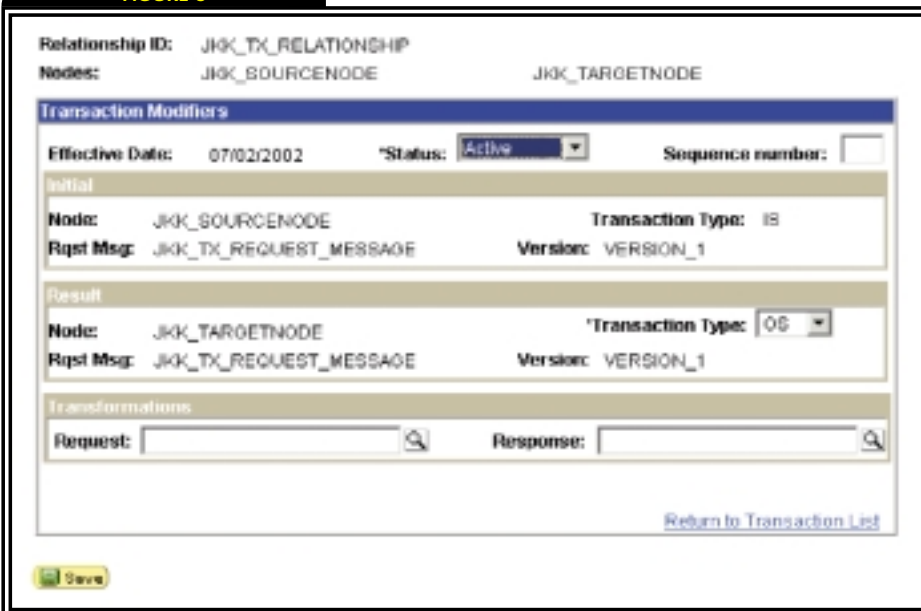
JMSUrl specifies the host and listening port of the target JMS server. In this case, the server is WLS 7.0-based and resides at http://wls01:7001. JMSUserName and JMSPassword are valid

FIGURE 4



Target node with associated JMS target connector

FIGURE 5



Source-to-target relationship definition

BEA dev2dev Days  
[www.bea.com/events/dev2devdays](http://www.bea.com/events/dev2devdays)

user ID/password combinations on the target WLS 7.0 instance. JMSFactory and JMSTopic reference the factory and topic defined within the target JMS server's config.xml file. A JMS listener or message-driven bean (MDB) listening to the same combination of JMSFactory, JMSTopic, and JMSServer will rendezvous with the IB and receive any messages transmitted by the broker via JKK\_TARGETNODE.

The next step is to define the necessary inbound and outbound transactions and associated relationships. IB transaction mappings must be made from the perspective of the PeopleSoft instance's default local node. Hence, an inSync transaction conveying JKK\_TX\_REQUEST\_MESSAGE and returning JKK\_TX\_RESPONSE\_MESSAGE must be defined at JKK\_SOURCENODE, while an outSync transaction with identical message characteristics must be defined at JKK\_TARGETNODE.

In addition, a relationship, shown in Figure 5, will be created to tether the two nodes together so inbound message traffic is routed to the target node and ultimately back to the WLS instance via the JMS target connector. Relationships, like many PeopleSoft objects, can be activated or deactivated, and can even become applicable upon a given effective date. All of this provides considerable flexibility, as multiple relationships can be defined across a set of node and date combinations.

In addition, relationships can be used to reconcile differences in the data formats exchanged between a combination of source and target nodes. For example, the data emanating from JKK\_SOURCENODE may need to be transformed, translated, or even filtered en route to JKK\_TARGETNODE. A relationship can perform these data manipulations using PeopleSoft batch programs called *application engine programs*. These programs can use a combination of PeopleCode, Extensible Stylesheet Language Transforms (XSLT), or SQL actions to perform the required data manipulations between a given combination of nodes. Such programs would be specified in the Transformations section of Figure 5.

*“By eliminating the need for client software, PIA's comprehensive Internet accessibility enables customers to realize substantially lower deployment costs than were previously possible under traditional client/server architectures”*

The next step is to configure the Integration Gateway to use the appropriate inbound JMS topics and queues. This configuration information is stored in the integrationGateway.properties text file, which is located under the Integration Gateway subdirectory of the installation tree. The gateway can listen to any combination of JMS topics and queues, and can use different JMS providers. For this example, the WebLogic JMS provider will be used, along with a sin-

gle inbound JMS topic and a special gateway error topic. The gateway will publish processing errors to this topic, in addition to writing the errors to a cumulative error log file. This design would accommodate a custom error topic subscriber that could handle gateway errors programmatically.

Listing 2 shows the portions of an integrationGateway.properties file relevant to the integration example. The ig.isc.serverURL, ig.isc.userid, ig.isc.password, and ig.isc.toolsRel properties specify the Jolt address, along with a valid user ID/password combination for the target application server. These properties effectively connect the gateway, which resides on the Web server, to the appropriate application server. Messages that enter the gateway will, under normal conditions, be routed to the target application server for processing by PeopleCode message subscriptions and OnRequest handler programs.

The ig.jms.JMSProvider.JNDIFactory.WebLogic property setting instructs the gateway to use WebLogic as the JMS provider. Next, the ig.jms.Topics, and ig.jms.Topic1 lines define the total number of JMS listening topics and specific topic connection information, respectively. The subsequent ig.jms.ErrorTopic lines specify the JMS error topic to which gateway errors should be published. The JMS connection properties shown in Listing 2 will cause the gateway to rendezvous with topic publishers and subscribers on the WLS instance server.

The final step of this example is to create the actual JMS topic publishers and subscribers that will rendezvous with the gateway via the topics defined in the integrationGateway.properties file. Specifically, a JMS subscriber will be needed to cover both the error topic and the PostingStatusMessage topic specified in the JMS target connector. These JMS subscribers can be standard JMS message listeners or even MDBs. A standard JMS topic publisher can be used to send JMS TextMessage instances into the gateway.

Listing 3 shows three message properties that must be set prior to publishing JMS TextMessage instances to the gateway. The first property, MessageName, specifies the message associated with the textual message being published. The DestinationNodes property specifies the default local node for the target PeopleSoft system. Finally, the RequestingNode indicates the IB node sending the published messages. Since the MessageName and RequestingNode shown in the listing match the inSync transaction previously defined at JKK\_SOURCENODE, the broker will accept and route the published messages to the JKK\_TARGETNODE node.

### Conclusion

The concepts, definitions, and procedures outlined in this article enable J2EE and PeopleSoft 8.4 applications to be seamlessly integrated using JMS. The benefits of taking this approach are clear: standards-based protocols and data (e.g., JMS and XML), transactional integrity, assured message delivery (both within PeopleSoft IB and WebLogic JMS), and high scalability using a combination of mechanisms such as message channels and pools of MDBs. Another benefit is that it can be used to integrate J2EE with any PeopleSoft 8.4-based product, including Financials, HRMS, SCM, and CRM.

It's important to remember that this article provides an outline, not an exhaustive treatment of this topic. Important issues such as security have not been discussed. Others, such as transaction management, are only implied by virtue of using JMS, but have not been covered. Both J2EE and PeopleSoft's Integration Broker are vast, complex technologies that cannot be covered in any appreciable detail within the constraints of this article. However, this outline can become the first step in the journey toward mastery of these technologies. The next article in this series will explore the integration of J2EE and PeopleSoft 8.4 applications using Web services. 🍌

Covasoft  
www.covasoft.com/today's\_tech/



Why are application servers so boring? I guess the answer to this question depends on your perspective. One man's boring commodity is another man's lifeblood. That observation alone would make for a rather short column, so we need another question... Who are these men, anyway?

## WebLogic Server System Administration

HERE'S WHERE THE EXCITEMENT BEGINS

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

Application servers are usually viewed from the perspective of an application developer. After all, J2EE is a lovely, thick set of specifications; they're very attractive to those of the developer persuasion, and it looks good if you can spell the acronyms correctly on your resume. All in all, not boring at all. A must-have commodity, but this wasn't always the case.

The advent of the application server reflects the increasing maturity of software engineering as a discipline, as it moves from art firmly to science. Despite this maturation, it amazes me when I run across developers who still delight in writing their own application infrastructure. "Not for me the straightjacket of a managed environment!" they exclaim, happily pulling a thread-pooling library from their pocket and inserting it into their next software magnum opus. This behavior reminds me of the book *Round Ireland With a Fridge*, whose intrepid author takes a fridge hitchhiking around Ireland. It makes for very amusing travel writing, but I would bank on it's not being the most effective mechanism of fridge transportation ever conceived.

Many application developers also find system administration boring. After all, that's the tedious

part of an application's life; the exciting, creative part is over, and all that's left to do is sit and wait for the little red light to flash, whereupon you reboot the server. How boring is that?

Although it's a rhetorical question, this deserves an answer. If you own the budget for an application, then its operational lifetime is not at all boring. Far from it. Bear in mind that a useful application is likely to have a production lifetime of more than 10 years. Now, compare that with the development cycle, which in these days of application integration and component reuse typically comes in at six months or less. Imagine you could save \$1 for every day of this period.... Now what's boring? If I can pocket the savings, I'll gladly take the 10-year operational lifetime and claim my \$3,650 - the development cycle is hardly interesting at all.

This observation leads wise users to look beyond the licensing cost of a software product suite and shift their focus to the total cost of ownership over the solution's deployment life cycle. It's also what led those mature software engineer types to put away their customized thread-management libraries and take a long, hard look at application servers. Why? Well, every customized component of a runtime engine is another piece of infrastructure that needs care and feeding in deployment from a person trained in its idiosyncrasies. That's bad for the organization that has it deployed - what if their pet guru goes on vacation, or worse, falls under a bus? It's also bad for the gurus. All that knowledge they have to build up isn't transferable, so career prospects outside their favorite machine room or away from the beloved company pager (presumably an engraved gold pager, since they've been there so long) are limited. How much better were they application server administrators (or better still, WebLogic Server administrators!)?

What does this have to do with transactions? Well, it's all very sexy thinking about two-phase commits and ACID (if you're so inclined), but at the end of the day, transactions are about the integrity of your valuable business data. What most operations staff lose sleep over is losing valuable business data. For this reason, they need tools to monitor what's going on, to make sure all is healthy or - if the worst happens - to work it out when things start failing in production so they can get systems back on track with minimal fuss and data loss.

What tools does WebLogic Server provide to

# Precise Software Solutions

[www.precise.com/wldj](http://www.precise.com/wldj)





enable the system administrator to see what's happening and get things back on track after an outage? What administrative capabilities are provided by WebLogic Server with respect to transactions? Let's take a whistle-stop tour of what's in the box for the professional WebLogic Server administrator.

### Server Administration

I always thought that in any distributed system it went without saying that the transaction logging and recovery services would be distributed. Apparently not, however, since one of WebLogic Server's major competitors centralizes the transaction log on a central administration server, providing a rather bizarre single point of failure. Rest assured, no such weaknesses are architected into WebLogic Server, so the first set of capabilities deals with managing distributed transaction logs. Every server instance that runs the JTA system has its own set of transaction logs. Ordinarily, they're cleared up by the server when they're no longer current, but if a server fails, decided transactions (written in the transaction logs) need to be rolled forward. This happens automatically if the associated WebLogic Server instance is restarted; if that's not possible, the logged transactions need to be recovered by another member of the cluster. Migrating the transaction recovery service from the failed server to a backup does this. In order for the recovery to be effected, the transaction log files must be visible to the backup server (since they are processed by the recovery process), so a dual ported disk, SAN, or other such technology must be in place. When the original server is rebooted the recovery service will gracefully return there, if the recovery has completed.

If the failure is too serious for an automatic recovery (head crash on the database, fire in the cabinet, etc.), an undocumented utility will dump out the contents of the transaction log in readable format. This could prove valuable for diagnosis of serious problems, and its details should be obtainable through BEA support.

Regular readers of this column will by now be true fanatics of the heuristically completed transaction. For such folk, two goodies are in the administrator's kit bag. First, the administrator can set a transaction abandonment timeout. If a transaction can't be completed before this limit is reached (maybe because a database isn't present), the server will give up trying. Since data may now be inconsistent, one of our beloved heuristic outcomes has occurred. But look on the bright side, the server didn't waste an infinite amount of energy trying to achieve the impossible. Of course, as in all these situations, the server will write a log record to record the failure.

The second place where heuristic outcomes may occur, and are logged, is in the heuristic log. If the server has imported a transaction from a foreign transaction manager (e.g., via OTS over IIOP), the server itself acts pretty much like an XA resource. This gives it the right in rare catastrophic situations (for example, after the transaction abandon timeout expires or if the XA resources participating in the WebLogic Server imported transaction throw heuristic exceptions) to make a heuristic decision. That is, unilaterally roll the transaction forward or back without the foreign transaction manager's say-so. When this happens, the event is recorded in the heuristic log file. This coexists with the tlog files, with a .heur in the filename before the .tlog extension. Look here when things go wrong in very convoluted deployments. (I recommend that you only look here after a stiff drink – in this kind of situation, you could be in for a long night!)

In less urgent times, the administrator can monitor transactions as they pass through the system. Clearly, since transactions should be of short duration, this will be an ever-changing snapshot of the system state, or it should be. One way an administrator can get early warning of impending disaster is if transactions start taking a long time to complete. For this reason, it's possible to sort the display of active transactions by how long they've been active. Application developers need to give the administrator guidelines as to how things should behave under normal conditions, but the sudden appearance of many long-duration transactions could be an early warning of a bigger problem.

While we're talking about the need for communication between developers and operators, we should review a few more features. WebLogic Server allows a name to be associated with each transaction; this can be done programmatically through the `weblogic.transaction.Transaction` interface or automatically by the EJB container for container-managed transactions. When the administrator is looking at the active transactions, or a dump of the transaction log, knowing the business operation that was being attempted in the scope of the transaction (developers: give them meaningful names!) could well give a starting point to fault investigations. For example, the account transfer transaction will touch two databases, the update address details will access the customer database, and so on.

WebLogic Server administrators can also set limits in the server: the maximum allowed number of concurrent transactions, the default transaction timeout, the abandonment timeout, and so on. In application server configuration in general, and in JTA in particular, it's important to use these limits to throttle the amount of work the application server will undertake. Bear in mind that for a deployment on any given set of hardware, there will be a physical limit to the possible throughput. By using administrative throttles to prevent the application server from trying to cross this physical threshold, you can ensure that your deployment will disappoint a few users while it quiets down from operating at the limits, rather than disappointing all the users together when the whole thing grinds unceremoniously to a halt due to lack of hardware resources to continue running.

### Summary

Wow... I'm exhausted! I feel like that was a high-speed sprint through the foothills of a mountain of information. Inevitably, the techniques and nuances of management for any given application fall firmly outside the scope of a single, relatively short, generic column, but I hope this has provided a flavor of the facilities available and pointed you to where to go for more information.

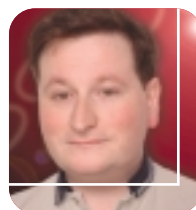
In conclusion, I'd like to confirm that it's true. If admin is boring, then application servers are boring (and WebLogic Server is more boring than most) because while application servers do offer good prospects for the developer, their true value is realized when they're sitting in the background doing what they're best at – providing years of trouble-free operation to the systems they support.

### References

- Hawks, Tony. (2000). *Round Ireland With a Fridge*. St. Martin's Press.
- *The WebLogic Server JTA Administration Guide*: <http://edocs.bea.com/wls/docs70/adminguide/managetx.html>.

# Sitraka

[www.sitraka.com/jclass/wldj](http://www.sitraka.com/jclass/wldj)



BY  
TAD STEPHENS & ERIC GUDGION

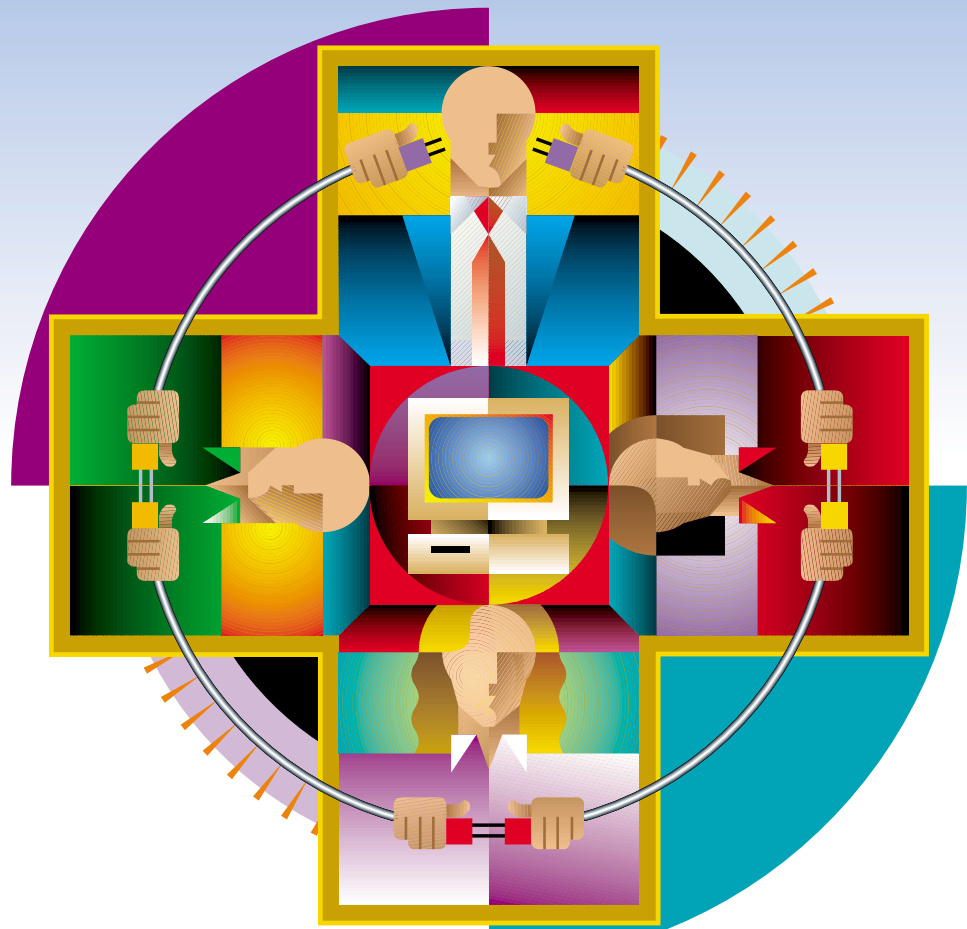
**AUTHOR BIOS...**

Tad Stephens is a system engineer based in Atlanta, GA, for BEA Systems. Tad came to BEA from WebLogic and has more than 10 years of distributed computing experience covering a broad range of technologies, including J2EE, Tuxedo, CORBA, DCE, and the Encina transaction system.

Eric Gudgion is a principal systems engineer with the Technical Solutions Group. His background in mainframe systems comes from years of work within the field as a system programmer and system engineer for various mainframe vendors.

**CONTACT...**

tad@bea.com  
eric.gudgion@bea.com



# WebLogic Server on the Mainframe PART 4

## STRATEGIES FOR YOUR BIGGEST CHALLENGE

Integration is the biggest challenge facing IT organizations. The glass house controls much of the business-critical data in the enterprise, and traditional integration tactics, while complex and often proprietary, are still king. However, more convenient, aggregated, and flexible access to this data is requested and often required by the business side to extend, or at least defend, competitive advantage.

Customers and users expect more access, better services, and more current data, and a company's competitors are just a click away. Fortunately, products such as the BEA WebLogic Platform, industry standards including Web services, and deployment of WebLogic Server on the mainframe provide a bridge over these "troubled waters." Deploying WebLogic Server on the mainframe allows access to legacy data and applica-

tions to be opened to Web services as well as standards-based adapters and interfaces.

This is the final article in our series covering WebLogic Server on the mainframe. The first article detailed many of the business advantages of deploying J2EE applications within the mainframe. These benefits include leveraging Java for better programmer productivity; aggregating multiple servers onto a single mainframe partition to lower operational costs and more efficiently utilize existing hardware; leveraging mainframe quality-of-service capabilities for 24/7 application availability; and extending existing applications and data located on the host machines. The second article detailed how to install and configure WebLogic Server for z/Linux and z/OS environments, including the steps required, the resources necessary on the mainframe, and how this differs from installing WebLogic on other platforms. In the third article

Rational Software  
www.rational.com/offer/bea

we covered many of the tips and tricks we learned with WebLogic Server in z/Linux and z/OS environments and provided an overview of several customer production deployments.

This month we'll outline the various integration strategies for accessing mainframe resources, applications, and databases, including:

- Using WebLogic Server on the mainframe to Web service-enable mainframe data and applications
- Utilizing WebLogic Workshop to build Web services applications efficiently, including an example from a third-party product
- Leveraging the various capabilities of WebLogic Integration
- Accessing MQSeries or other messaging technologies from WebLogic Server

### Enabling Mainframe Applications with Web Services

One of technology's most popular buzzwords is Web services. A key advantage of Web services is the use of a standards-based interface to expose components, objects, and services to programmatic clients, regardless of their location, underlying hardware, or language in which the program was written. The promise of Web services is particularly attractive for the mainframe, since the majority of these applications are written in legacy programming languages such as COBOL, and offer limited integration alternatives. Opening the door to these applications via Web services provides a powerful, more flexible, more current programming paradigm. However, the challenge is in providing this promising technology on the mainframe.

By offering a complete implementation of Web services, as well as deployment support to z/Linux and z/OS mainframe environments, WebLogic Server offers a unique platform that spans these technologies. On the Web services side, WebLogic Server offers a complete implementation of Web services standards, including SOAP, WSDL, UDDI, HTTP, and XML, as well as enterprise features for loose coupling, asynchrony, secure authentication, authorization, and message encryption. Utilizing the power of the Java environment for platform independence, the same WebLogic Server that runs on Unix, Linux, and Windows also runs on the mainframe and includes support for Web services. The administrator is free to deploy his or her application on the hardware platform that best suits the business requirements without affecting the development effort.

However, deploying WebLogic Server in z/Linux or z/OS doesn't magically expose Web services within the underlying business application. WebLogic Server provides a Web services listener in a Java Virtual Machine executing on z/Linux and z/OS operating systems. Business application environments and databases such as CICS, DB2, or IMS, still require access via traditional adapters, client APIs, or interfaces. WebLogic Server offers this proxy-style capability, exposing a Web service interface to SOAP clients and mapping the corresponding WSDL to the mainframe application. When deployed on the mainframe, the Web service proxy, adapter, and application itself are consolidated onto a common platform, simplifying development, deployment, and management.

### Mainframe Web Services with WebLogic Workshop and NEON ShadowDirect

BEA recently introduced a powerful tool to aid developers building Web services-based applications: WebLogic Workshop. WebLogic Workshop provides a graphical environment for developing, assembling, and deploying applications using Web services and J2EE. One key use of this tool is in the assembly of complex

business components, tying everything together with Java code. The components may include EJBs, database objects, or other Web services. In addition, WebLogic Workshop includes an open API for representing custom connectors. One example is the ShadowDirect J2EE Connector Architecture (J2EE CA) adapter from NEON Systems, Inc.

ShadowDirect enables WebLogic Server to connect, using J2EE CA or JDBC, to a variety of mainframe resources, including:

- CICS
- IMS/TM
- IMS/DB
- DB2
- VSAM
- ADABAS
- Natural
- Flat files

In addition, the ShadowDirect adapters can leverage mainframe services such as Resource and Recovery Subsystem (RRS) and Workload Manager (WLM) if required, although these aren't prerequisites.

Use of the J2EE CA adapter incorporates the J2EE component paradigm for reuse, application design, and simplified maintenance. The ShadowDirect adapters handle the complexities of communication to the data source while the underlying J2EE architecture dispatches the request to an available pooled connection. WebLogic Server provides connection pooling, high availability, fail-over, clustering, session persistence, and support for Web services.

Here's where things get interesting. WebLogic Workshop is a powerful tool for building Web services applications, including those that make use of the ShadowDirect adapters. With access to a few lines of Java code and the ShadowDirect controls for WebLogic Workshop, a J2EE programmer can easily wire together complex components that represent mainframe data and applications. In addition, these programmers need no mainframe skills or knowledge of the underlying complexities. WebLogic Workshop generates the code needed to access the ShadowDirect adapters from any Web services client that supports SOAP and WSDL, including Microsoft .NET. Given the diminishing number of programmers trained for mainframe programming, the ability to leverage Java developers can provide an attractive return on development investment.

NEON Systems provides an online example (<http://houscbeawl.neonsys.com/webservices.html> and <http://houscbeawl.neonsys.com>) to illustrate how ShadowDirect can be used to access the mainframe. You'll find an overview of the example application, the corresponding configuration for building the sample, and a test harness for validating the example application. The service descriptions represent the services available on this test application. All mainframe access is implemented using ShadowDirect. A key advantage is that all the data sources are defined and accessed using well-defined XML documents.

Figure 1 presents the WebLogic Workshop design environment and the controls for accessing the mainframe via ShadowDirect. Using the Design view, the developer can quickly and easily arrange components that represent mainframe services.

### WebLogic Integration and the Mainframe

A key component of the WebLogic Platform offering is the WebLogic Integration product. WebLogic Integration combines an adapter suite based on the J2EE CA, a robust business process management service including data translation and transforma-

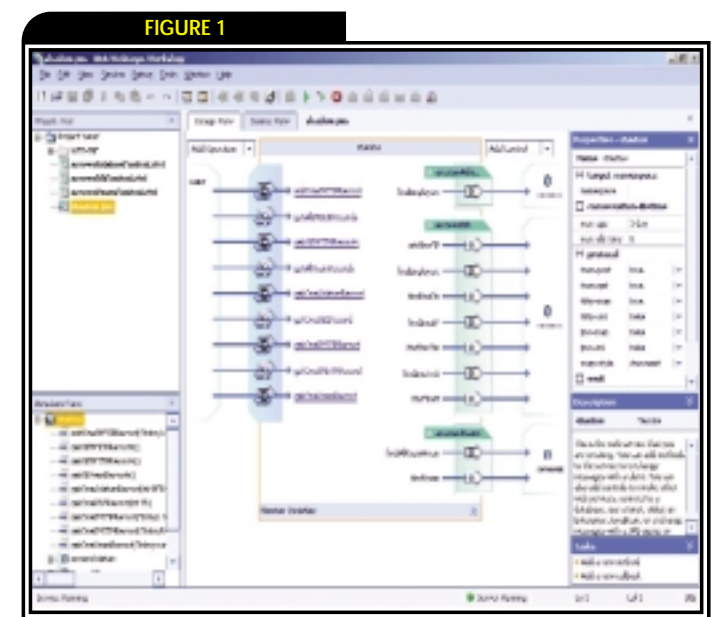
tion services, and a B2B engine that facilitates communication between organizations. Hosting this product on the mainframe offers a number of benefits:

- Unified platform of application server and integration engine.
- Consistent administration and management.
- Adapters are closer to the data and applications they access, decreasing the connections and corresponding network latency.
- All applications are hosted within a single mainframe operating environment.
- The Java development and programming model is insulated from the underlying mainframe environment.
- Reliability, scalability, and availability are achieved through the combined benefits of the mainframe operational support and WebLogic Server.

Leveraging industry standards such as J2EE CA and Web services, WebLogic Integration is able to deliver solutions more quickly by reusing infrastructure. Standards lower the risks associated with technology decisions, lower maintenance costs through a platform that will evolve as technologies change, lower time-to-market through reusable services, and lower training and administration costs.

WebLogic Integration provides a complete business process management engine, including support for business process modeling, process execution, and process monitoring. In addition, it includes a full suite of routines for data translation and data transformation, as well as a plug-in framework where new actions, functions, variable types, and events can be registered. WebLogic Integration supports event processing, providing support for both automated and manual processing. For example, the ability to allow manual intervention in a business process allows our solution to model real business problems. It is designed to be extensible, adapting to unique business problems and scenarios.

A suite of services that enable businesses to connect and collaborate with business partners, including execution of transactions, automation of business processes between enterprises, and management of ongoing B2B relationships, is also included. It supports a broad range of connectivity options, from simple browser clients and basic file sharing to EDI and RosettaNet exchanges. Business



WebLogic Workshop design environment

message and transaction support, including EDI, XML, RosettaNet 1.1 and 2.0, cXML, and XOCF, allows trading partners to engage in long-running, high-volume, complex business transactions. Enhanced security features include support for digital signatures, nonrepudiation, and trading partner agreements.

### Integrating WebLogic Server with Other Integration Tools

The mainframe is a mature environment, and a wide variety of technologies exists for access to host-based applications and data. Many products provide proprietary client APIs. When considering WebLogic Server on the mainframe, access to these APIs is possible as long as a Java-based client API exists. The primary advantage is that applications developed using Java and J2EE can now leverage access to these messaging systems and brokers with few or no changes required. One example is access for foreign JMS providers, such as MQSeries, using the JMS APIs. (The use of the term "foreign" simply implies the JMS provider is external to WebLogic Server.)

Although WebLogic Server includes a complete JMS 1.0.2 service implementation, it can also be used with foreign JMS providers such as IBM's MQSeries, SonicMQ from Sonic Software, SmartSockets from Talarian, FioranoMQ from Fiorano, and Ibus Message Server from Softwired. WebLogic Server can interface with these messaging systems and can consume messages from these systems via the message-driven EJB, or MDB. BEA provides a white paper (<http://dev2.dev.bea.com/resourcelibrary/whitepapers.jsp?highlight=whitepapers>) detailing how to interface these systems via the MDB, using the WebLogic messaging bridge, and in a standalone client program. In addition, code examples and a working MQSeries sample application can be found at <http://dev2dev.bea.com>.

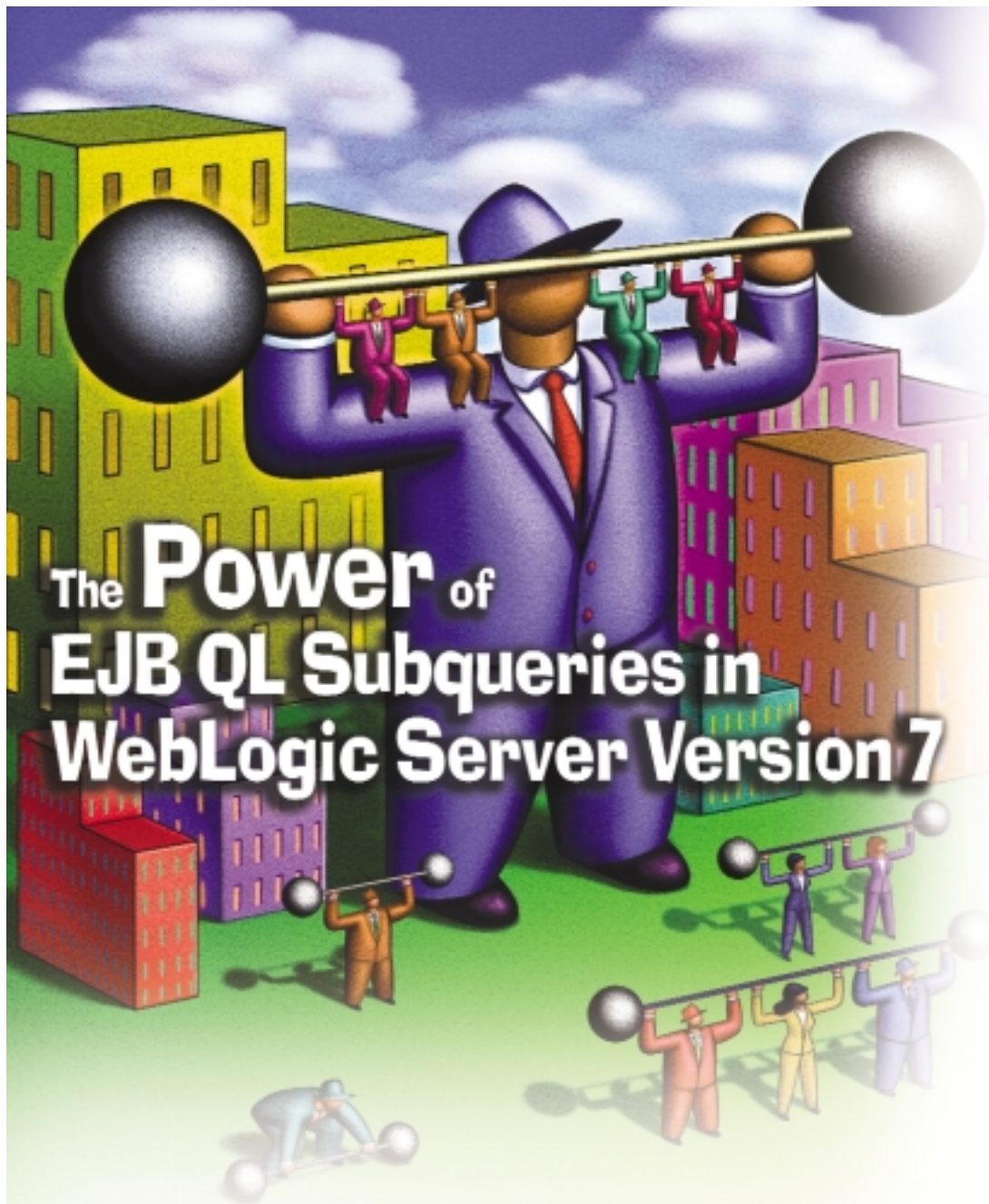
Let's consider an example interfacing with MQSeries. The typical usage pattern is to define a WLS MQSeries XAConnectionFactory object that wraps an MQSeries XAConnectionFactory object. This WebLogic Server-specific connection factory might then be stored in a WLS JNDI context for use at application runtime. An application would retrieve the connection factory and use it to create the appropriate connection, session, producer, and consumer objects. Each of these objects derived from the connection factory will be a WLS MQSeries object that delegates method invocations to the corresponding MQSeries implementation object.

Similar approaches can be taken with other integration brokers and systems provided a Java client API exists. The key advantage of WebLogic Server is that it can be deployed on the mainframe along with the specific integration services, providing a single hardware environment hosting the end-to-end solution. The resulting solution is easier to manage, and performance and reliability problems related to network latency are avoided.

### Summary

We've tried to show how easy it is to leverage WebLogic Server on the mainframe. There are a number of areas where costs can be reduced, hardware resources can be better utilized through server consolidation, programmer productivity can be increased, and investment in existing systems can be leveraged. The installation and configuration assistance will aid anyone deploying WebLogic Server on the mainframe, and the tips and tricks will aid in performance tuning and analysis. Tighter integration of data and applications results in a simpler, easier-to-manage environment. J2EE running on the mainframe is a powerful solution and can solve a broad range of business problems facing today's IT manager.





BY  
THORICK CHOW

#### AUTHOR BIO...

Thorick Chow is an engineer at BEA Systems. After receiving his degree in physics from the University of California at Berkeley in 1988, he joined the client/server revolution at Sybase. In 1998 he joined the WebLogic team and has been involved in the development of WebLogic JDriver and WebLogic EJB products.

#### CONTACT...

thorick@bea.com

The standard EJB 2.0 container-managed persistence (CMP) query language known as EJB QL allows users to retrieve container-managed entity beans, subject to constraints that are described using the same object-relationship model that was constructed to describe beans in EJB deployment.

With this language you can reduce large classes of search problems to the task of writing queries that can be expressed in a straightforward and natural manner.

Some classes of problems can't be solved using the capabilities of standard EJB QL, however. One

such class can be described intuitively as "that class of problems for which a single scan of the retrieval candidates isn't enough to narrow down the beans that qualify to be selected." For example, suppose you have a set of beans that represent recordings. You want to find the set of recordings whose sales totals are above the average taken of all recordings. This isn't possible looking through the candidate data just once, because in order to know which recordings have above-average sales totals, you must first know what the average value is.

Answering this kind of question using EJB QL requires something extra. WebLogic 7 gives you that something extra with support for subqueries in EJB QL. This article will demonstrate different ways to harness the power of this new feature to

answer previously unanswerable questions using EJB QL.

The example queries use the EJB CMP beans and relationship schema defined in the "Bands" example, which can be found in WebLogic Server Version 7 at `RELEASE_DIRECTORY\samples\server\src\examples\ejb20\relationships\bands`.

The Bands example contains four entity beans: BandEJB, RecordingEJB, FanClubEJB, and ArtistEJB. Figure 1 illustrates the relationships between these beans.

### The Structure of EJB QL Subqueries

As the name implies, a subquery is a query that's subordinated within an outer query. As EJB QL is an SQL-like language, the WebLogic EJB QL subquery (EJB QL subquery) syntax resembles that of subqueries used in SQL. As in SQL, the EJB QL subquery is used as an operand within the WHERE clause of an outer query as a producer of values to be used by an operator.

Suppose you want to write the EJB QL for a finder that will return all the BandEJBs that meet the following criteria: for the years 1961 and later, find the bands that have released a recording that has sold more copies than the average number of copies per record sold of all recordings.

This type of search criteria can't be expressed using standard EJB QL because there's no provision for computing the average of the number of copies sold, nor is there a provision for comparing a particular recording's copies sold with the computed average. This query can be written utilizing the WebLogic EJB QL language extensions *support for aggregate functions* (AVG or Average in this case) and *support for subqueries*.

Listing 1 illustrates an EJB QL query that will evaluate the search criteria. (Listings 1-9 are available for download at [www.sys-con.com/weblogic/source.cfm](http://www.sys-con.com/weblogic/source.cfm).) There are a few things worth noting about this query:

- The subquery's SELECT statement "SELECT AVG(subquery\_records.numberSold)" makes use of WebLogic EJB QL language extension support for aggregate functions, the AVG() (i.e., average) function in this case. The subquery will return a single scalar value - the average of the numberSold field of the quali-

fied subquery\_records qualified by the subquery's WHERE clause.

- The subquery (all the text between the enclosing parentheses in the first WHERE clause) is syntactically identical to a "normal" query. You could cut and paste the subquery and use it as the text of a complete query in itself (in an ejbSelect that returns the average number of units sold per recording, for example).
- The identifier variables declared in the subquery's FROM clause are distinct. That is, the subquery range identifier variable "subquery\_band" is distinct from the corresponding main query range identifier variable "band". Likewise, the subquery collection member identifier variable "subquery\_records" is distinct from the main query collection member identifier variable "records". All the identifiers declared in all FROM clauses in a query and its subqueries must be unique.

The subquery selects all recordings recorded by all bands after December 31, 1960. The average is taken of the numberSold of all the recordings selected. The main query then qualifies a band for selection by requiring that a candidate band has a recording recorded after December 31, 1960, that has sold more copies than the average value previously computed by the subquery. The collection of qualified bands comprises the results of the query.

### Correlated and Uncorrelated Subqueries

The query in the previous example is an uncorrelated subquery. An uncorrelated subquery can be thought of as completely self-contained. All identifier variables in an uncorrelated subquery are scoped within the subquery. A subquery that contains references to identifier variables declared by a parent query is referred to as a correlated subquery. For example, consider a query to return RecordingEJBs that meet the following criteria: find the records that are in the top-three ranking by number sold; show these top three records in descending order.

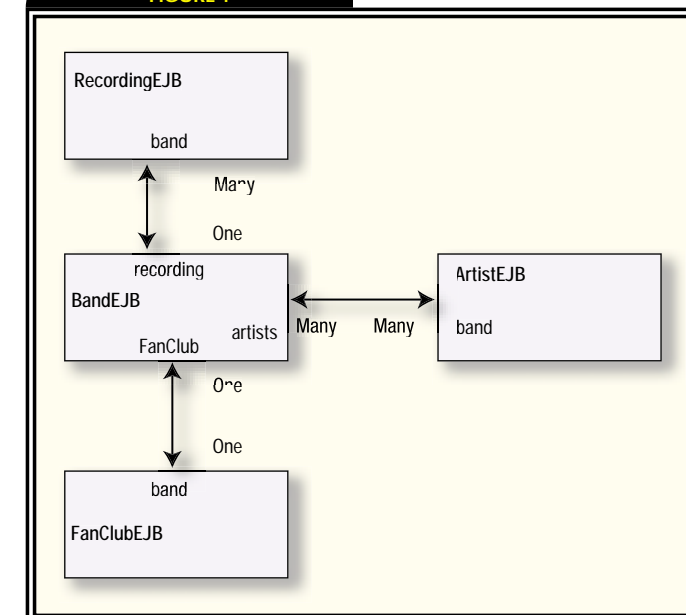
Listing 2 illustrates an EJB QL query that will return RecordingEJBs that meet this criteria. A few things about this query are worth noting:

- In the subquery's WHERE clause, the ">" operator's left-hand operand "subquery\_record.numberSold" contains the range identifier variable "subquery\_record", which is declared in the subquery.
- In the subquery's WHERE clause, the ">" operator's right-hand operand "record.numberSold" contains the range identifier variable "record", which is declared in the outer main query. This reference to a variable in the range of the outer query categorizes this subquery as a correlated subquery.
- The main query uses the WebLogic EJB QL language extension "ORDERBY <path-expression-ending-in-cmp-field> DESC". The ability to specify descending order in an ORDERBY clause is a new feature in WebLogic 7.

Each RecordingEJB is considered to be a candidate by the main query. The value of "record.numberSold" for each candidate is passed down to the subquery. The subquery does its own scan of all the RecordingEJBs, counting the number that have a value "subquery\_record.numberSold" that exceeds the value of "record.numberSold" passed in by the main query. If this number is less than three, there are at most two records in the entire set of RecordingEJBs that exceed the candidate. The candidate is in the top three, the the condition in the main query WHERE clause evaluates to "true", and the main query SELECTs the candidate.

This process is repeated by the outer query for each candidate

FIGURE 1



"Bands" example relationship schema

until all the candidates are checked for qualification. The top three RecordingEJBs are then returned in descending order by numberSold.

This query is processed in a different manner than the previous example's uncorrelated subquery. Unlike the uncorrelated subquery in Listing 1, which is evaluated only once, this example's correlated subquery (shown in Listing 2) will be evaluated once for each row for which the outer query produces a value "record.numberSold". Since the correlated subquery is evaluated separately for each candidate produced by the main query, the performance cost of evaluating correlated subqueries is something to keep in mind when making decisions about using them.

TABLE 1

EJB QL Operator	EJB Compiler Actions
IS NULL	May generate no subquery May generate an uncorrelated subquery May generate a correlated subquery
IS EMPTY	Will generate a correlated subquery
NOT MEMBER OF	Will generate an uncorrelated subquery or a correlated subquery

EJB QL Operators

### Subqueries Returning Multiple Values

In each of the examples we've considered so far, the subquery always returned a single scalar value. The operator that took the subquery as a right-hand argument in both cases was the comparison operator "greater than" (">") operator. This accepts only scalar values from its right-hand operands. If you try to use a comparison operator like ">" with a right-hand operand that returns multiple values, a runtime error will be thrown by the underlying SQL engine.

Consider a query to return BandEJBs based on the following criteria: find the bands that have released any recording that has sold more units than the number of units sold of any recording before 1961.

Listing 3 shows one attempt at writing this query. In general, this query won't work – it's very likely that there was more than one recording recorded before 1961, and the ">" operator wouldn't be able to handle the resulting multiple-valued right-hand argument.

If Oracle were used as the underlying SQL engine, you'd receive the runtime SQL Exception: "ORA-01427: single-row subquery returns more than one row". Similarly, SQL Server would return the corresponding runtime SQL Exception: "Server: Msg 512, Level 16, State 1, Line 1 Subquery returned more than 1 value". This isn't permitted when the subquery follows =, !=, <, <=, >, >=, or when the subquery is used as an expression.

To fix this, use an operator that expects the subquery to return multiple values. In this case, follow the ">" operator with "ALL". The corrected query is shown in Listing 4.

This is an uncorrelated subquery, so it is evaluated first to find the set of all records recorded before January 1, 1961. The main query then considers a candidate band and a candidate recording from the candidate band. If the recording.numberSold exceeds the numberSold of the set of ALL records selected by the subquery, the main query SELECTs the band.

All the comparison operators {=, <>, <, >, <=, >=} expect their right-hand arguments to be single valued. If the subquery argument to any of these operators may return more than a single value, the appropriate qualifiers "ALL" or "ANY" should be appended after the comparison operator.

Subqueries that may return multiple values can also be used

with the [NOT] IN and [NOT] EXISTS operators. To illustrate the use of [NOT] IN, let's suppose the legal department of a recording company wants to answer the following: find the names of all bands such that each returned band is named after the founder of a different band and the founder of the different band is not a member of the returned band.

Listing 5 illustrates a query that answers this question using the [NOT] IN operator in combination with a correlated subquery. Let's consider the workings of this query in detail. An examination of the FROM clause of the outer query reveals three separate range variable declarations – two that refer to the BandEJB and one that refers to the ArtistEJB. Two separate declarations for the BandEJB are required so we can distinguish between the two sets of bands that need to be compared. The identifier targetBand represents a band considered for inclusion in the final results, and the identifier founderBand represents a band whose founder a targetBand will have taken as its namesake. The identifier founderArtist represents the artist who is the founder of the founderBand. For example, suppose that the founderBand is "X" and that the name of the founder of X is "John Doe." The founderArtist will be the ArtistEJB representing John Doe, the founder of the founderBand X. The task of the query, then, is to locate any band named John Doe that doesn't include the artist named John Doe as a member.

At execution time, the outer query chooses a candidate band subject to criteria specified by its join between the targetBand and the founderBand: the name of a targetBand must be equal to the name of the founder of the founderBand John Doe. The subquery then takes the candidate band John Doe presented by the outer query, and selects the set of the IDs of all artists that are members of the band John Doe. The outer query checks whether the founderArtist ID for John Doe is [NOT] IN the set of artists that are members of the band John Doe that the subquery

"An uncorrelated subquery can be thought of as completely self-contained"

returned. If founderArtist ID for John Doe is [NOT] IN the set, the band John Doe is added to the set of bands that the main query will return. This procedure is repeated for all of the candidates the outer query chooses. The set of qualifying bands is returned as the final query result, and then the company legal department goes to work.

Consider the differences between the [NOT] IN operator and the [NOT] EXISTS operator. Where [NOT] IN is used to check whether its left-hand operand is contained within the set returned

# Sitraka

[www.sitraka.com/performance/dev2dev](http://www.sitraka.com/performance/dev2dev)



by its right-hand subquery operand, the [NOT] EXISTS operator has only a right-hand operand and checks whether the set returned by its right-hand subquery operand is empty or not. To illustrate the use of [NOT] EXISTS, Listing 6 shows the query of the previous [NOT] IN John Doe example rewritten to use the [NOT] EXISTS operator.

### Standard EJB QL and Hidden Subqueries

Let's consider again the [NOT] IN John Doe query of Listing 5. You may notice that the question that led to the formulation of this query could also have been expressed as a query using only standard EJB QL with the help of the NOT MEMBER OF operator. Listing 7 illustrates an equivalent query using NOT MEMBER OF.

It's interesting to examine the SQL the EJB compiler generates for the NOT MEMBER OF query. The complete generated SQL is shown in Listing 8.

Of particular interest is that the EJB QL NOT MEMBER OF operator was translated by the EJB compiler into a database query using the SQL [NOT] IN operator operating on a correlated subquery. The EJB QL NOT MEMBER OF operator is actually a hidden [NOT] IN operator that uses a correlated subquery. Now consider the EJB QL [NOT] IN John Doe query of Listing 5 that contained the explicit correlated subquery. The generated SQL for this query is shown in Listing 9. Comparing the generated SQL for the EJB QL NOT MEMBER OF query of Listing 8 against the generated SQL for the EJB QL [NOT] IN query of Listing 9 reveals that the SQL is essentially identical. Therefore, to the underlying DBMS, the two queries are identical. Clearly, the query can be expressed more compactly in EJB QL using the NOT MEMBER OF operator.

At times it may be useful to opt for the use of explicit subqueries, even when they don't seem necessary at first. Consider the following: using MS SQL Server version 7 and a small set of test data, SQL Server's query analyzer has interesting things to say about the SQL query generated for the EJB QL NOT MEMBER OF query (translated into the [NOT] IN query of Listing 9) versus the generated SQL query of the equivalent EJB QL query rewritten using [NOT] EXISTS with an explicit subquery in Listing 6. SQL Server will perform the following major operations to evaluate the two queries:

1. EJB QL Query using NOT MEMBER OF with generated NOT IN with subquery:
  - Five merges
  - Two clustered index scans
  - One sort

- One full table scan
  - Three clustered index seeks
  - One temporary table
2. EJB QL Query using explicit NOT EXISTS with subquery:
    - Four merges
    - One clustered index scan
    - Zero sorts
    - One full table scan
    - Three clustered index seeks
    - One temporary table

In comparing the list of operations that SQL Server will perform to evaluate the [NOT] IN query versus the [NOT] EXISTS query, note that the [NOT] IN query will require five merges versus the [NOT] EXISTS query's four merges, two clustered index scans versus one clustered index scan, and one sort versus no sort. While the actual performance of SQL queries depends on many factors, the difference in the tally of operations in the query execution plans chosen by SQL Server illustrates that one query might be a better performer than the other. Thus, it may be advantageous to rewrite an EJB QL query that uses an operator like NOT MEMBER OF to instead use an explicit subquery. It must be noted that there's no general rule stating that the use of [NOT] EXISTS is more efficient than using [NOT] IN. The actual determination of which query style will be the better performer on which DBMS and under what conditions is the province of SQL performance tuning. What's important to remember is that with support for EJB QL subqueries, you have choices. This is especially important when dealing with SQL queries that involve correlated subqueries, which can be heavy users of DBMS resources.

For reference, Table 1 lists the standard EJB QL operators for which the EJB compiler may generate SQL with hidden subqueries.

The SQL generated by the EJB compiler can be examined by compiling the EJB with the "keepgenerated" option as in: `java weblogic.ejb -keepgenerated build\std_ejb20_bands.jar`

The output JAR file "ejb20\_bands.jar" will contain generated Java source files for each EJB. For example the generated Java source file for the BandEJB will be: `BandBean_lya094_WebLogic_CMP_RDBMS.java`

This source file will contain the generated SQL for each finder and ejbSelect method run by that bean and can be examined with your favorite editor.

### Summary

The introduction of EJB QL subqueries in WebLogic 7 gives you the ability to write finder and ejbSelect methods that are capable of satisfying an enriched set of search criteria over standard EJB QL. Because the syntax of WebLogic EJB QL subqueries resembles the syntax of SQL subqueries, the learning curve associated with EJB QL subqueries is short and painless. In a manner similar to SQL, EJB QL subqueries may be used as operands of the operators `{ { =, <, >, <=, >= } [ANY | ALL], [NOT] IN, [NOT] EXISTS }`. In addition to answering questions that could not previously be answered, subqueries provide the programmer options with regard to custom tuning of standard EJB QL queries when those standard EJB QL queries make use of hidden subqueries. Subqueries are a handy addition to the EJB programmer's toolbox.

Enjoy the enhanced capabilities of WebLogic EJB QL with subqueries. •

# WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

**\$99**  
+ S&H  
Special Limited-time Price



Now Shipping  
**\$119**  
CD  
VALUE  
FROM JDJ

EVERY ISSUE OF  
**WSJ & JDJ**  
EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF  
EXCLUSIVE WSJ & JDJ ARTICLES ON ONE CD!

"The Secrets of the  
Web Services Masters"

CD is edited by well-known editors-in-chief

Sean Rhody and Alan Williamson

and organized into more than 50 chapters

containing more than 1,400 exclusive WSJ & JDJ articles.

[WWW.JDJSTORE.com](http://WWW.JDJSTORE.com)

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

MORE THAN  
**1400**  
EXCLUSIVE  
WEB SERVICES  
& JAVA  
ARTICLES

7  
YEARS  
83  
ISSUES  
1400+  
ARTICLES

ONE  
CD

ORDER  
ONLINE  
AT [JDJSTORE.COM](http://JDJSTORE.COM)  
SAVE  
**\$20**

**SAVE 16% OFF**

**ColdFusion** Developer's  
Journal

12 Issues for  
**\$89<sup>99</sup>**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

- Exclusive feature articles
- Interviews with the hottest names in ColdFusion
- Latest CFDJ product reviews
- Code examples you can use in your applications
- CFDJ tips and techniques

That's a savings of **\$17<sup>99</sup>** off the annual newsstand rate. Visit our site at [www.sys-con.com/coldfusion](http://www.sys-con.com/coldfusion) or call 1-800-513-7111 and subscribe today!

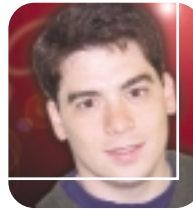


# Using JMX

MONITOR YOUR APPLICATIONS INTERNALLY AND EXTERNALLY

BY SAM PULLARA

The Java Management API (JMX) has been integral to managing the WebLogic Server since WebLogic 6.0. Through this API you can search for management beans (mbeans) within the application server and query them for both configuration information and runtime monitoring information. In addition, this API can be used to actually change the configuration of the server. In fact, this API is used internally by the console and other administration tools to do their work and report their data.



convenience behind the weblogic.Deployer class. Accessing them directly is possible but not recommended except for looking at configuration and not changing it. The rest of the attributes are fair game, although beware, most of those attributes will only actually change runtime behavior if the component (like a JDBC pool) is redeployed or if the server is restarted. Applications of these configuration mbeans range from determining the

resources that a particular application is using to tweaking values based on the feedback from the monitoring mbeans.

As you can see, there's a lot of information – but how do you get to it if you're new to JMS and the WebLogic Server? It's actually not that hard. As long as you can use JNDI and read JavaDocs you'll find it pretty easy to navigate the hierarchy of mbeans. In the code sample below you can see how you find the local server mbean home and then find the JDBC pool.

```
MBeanHome home = (MBeanHome) new
InitialContext().lookup(MBeanHome.LOCAL_JNDI_NAME);
Set pools = home.getMBeansByType("JDBCConnectionPool");
```

This gives you the configuration mbeans for the JDBC pools. If you want to monitor the pools rather than look at their configuration, simply change the type of mbean by adding "Runtime" to it. From this mbean you can determine important information like whether you are leaking connections – you can even get a stack trace of where exactly the connection was allocated!

What if you want to use the JMX mbean system without doing any Java programming, like from a script file? Fortunately, we include a tool that will let you view and change mbeans directly from the command line – weblogic.Admin. In addition to PINGing servers, getting JNDI listings, resetting pools, and shutting down the server, it gives you direct access to the mbean hierarchy. In the next code sample you'll see the equivalent weblogic.Admin command.

```
java weblogic.Admin -username system -password [password] -url [url] GET
-pretty -type JDBCConnectionPool
```

This gives you an easily readable description of the mbeans that match the type descriptor plus the properties on those mbeans. Similarly, there is a SET command that will change the properties on any mbean in the system.

Armed with this tool you should be able to monitor your applications internally in Java and externally from scripts.

This API can be a powerful tool for monitoring. Let's look at an example. In this application, you have mostly static content, some JSP pages for your personalized pages, a registration system, a JMS-based data feed, and maybe a small billing application. Where in this application can we use JMX to our advantage? On the static page side, you can access the information for the FileServlet and see how much time was spent serving the static content. It might be less than you expect because of the amount of caching that browsers do automatically on the client side. Most requests for static content will be HEAD requests, checking for changes. For dynamic pages, each servlet in the system is accounted for separately. This data doesn't perfectly reflect the user experience (because the user has the lag between when their request is sent and when they receive the page back), but it does tell just how long each servlet, on average, spends in a request and how many have been made. Our registration system is an entity bean, while the billing system is a stateless session bean backed by an external service. Both beans can be monitored and the information used to tune the size of the caches and pools. Our data feed can also be interrogated to determine if we have been getting as many messages as we have been paying for, or to determine how many subscribers are using the data. Throughput can also be calculated using the attributes on the JMS topic and queue mbeans. Finally, all JDBC connections can be monitored and from their mbeans you can find out how much the pool connections are contended, which will tell you if you need more database client licenses, for instance.

The second use of mbeans in the WebLogic Server is for configuration information. The configuration of all resources is contained at runtime by the configuration mbeans. These mbeans are essentially an instantiated representation of the information contained within the config.xml file. Everything from which applications are deployed to which port a server listens on is contained within these interfaces. Most of the individual attributes are relatively straightforward; however, the collection of attributes that describes a deployment unit is complicated and they have been collected for

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

Intel  
www.intel.com/ad/bea



# SUBSCRIBE TO THE FINEST TECHNICAL JOURNALS IN THE WORLD...



## IT'S JUST A CLICK AWAY!



SYS-CON MEDIA

WWW.SYS-CON.COM

# A Final Review

## CONCLUDING THE BEA WEBLOGIC SERVER 6.0 CERTIFICATION TEST STUDY GUIDE

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

WebLogic Server administration is the last area you need to study to pass the WebLogic Server 6.0 certification test. In this article, I'll cover some topics you may find on the test, such as deployment, security, and using the server console.

### Deployment

WebLogic Server supports auto deployment, configured deployment, and hot deployment. Hot deployment enables an EJB application to be redeployed, modified, or undeployed on an active domain. WebLogic Server performs these actions without requiring the server to be reset. Hot deployment is performed using the weblogic.deploy utility.

Auto deployment is hot deployment that's done automatically by WebLogic Server. The server monitors the application directory of a domain for new or modified applications. Once a changed application is detected, the appropriate files are hot deployed. If files are removed, WebLogic will undeploy the existing applications.

Configured deployment explicitly registers the applications in WebLogic Server. The administration console provides the mechanism to install a Web application and logs entries into the config.xml file, which informs the server which applications are to be activated. Configured deployment should be used in a production environment to avoid unexpected application deployments.

### Security

WebLogic Server supports a variety of security realms. A security realm is a logical grouping of security attributes. WebLogic provides the File realm, LDAP, NT, UNIX, and RDBMS security realms out of the box. Of these realms, the default security realm is the File realm, which uses security information from properties defined in the fileRealm.properties file.

You should know how to set up a Web application to use a security realm. Once the security realm is defined, users must log in to it to access the Web application. The security realm will define which resources the user may access.

To set up this mapping, the web.xml file must

first be updated with the <login-config> tag to indicate how users must log in. Next, web.xml must be updated with the security roles that exist in the Web application; do this by modifying the <security-role> tag. Using the <security-role-assignment> tag, modify the weblogic.xml file to map the security roles defined in web.xml to one or more principals in the defined security realm. Finally, to restrict access to specific resources based on a URL pattern, use the <security-constraint> tag of the web.xml file to map a resource to a security role.

### Server Console

Be aware of the server's behavior when booting it up. For starters, understand the concept of a domain. The config.xml file specifies the configuration attributes of a domain. This file should not be manually modified; use the administrative console to set the domain properties. As for the administrative console, be sure to familiarize yourself with the interface. There are a few questions on the test about specific parts of the administrative interface - logging, monitoring, and the impact of certain parameters. The best way to study for this is hands-on experience. At the very least, click through each of the screens in the console and read the Help file for each screen.

There are two ways to boot a WebLogic Server: administrative and managed. By default, the server will boot in administrative mode. In general, when you have multiple related WebLogic Servers running in a domain, only one should run in administrative mode. The other managed servers within the domain are administered by the sole administration server.

### JMX

Before I wrap up, there's one more topic you should look at: JMX. There aren't many questions on this topic, but you should devote some study time to it. As a refresher, JMX is a system-management specification to help monitor a WebLogic Server. Be sure to know the types of JMX beans that WebLogic Server 6.0 supports (see Table 1).

TABLE 1

JMX BEAN	DESCRIPTION
Administrative	Represents static server configuration
Runtime	Represents current statistics of a runtime resource
Configuration	Represents true server configuration, based on any overrides

Connection pool methods

### Conclusion

Well, that's it! You're now ready to pass the BEA WebLogic Server 6.0 certification test. Before you take it, though, be sure to complete the last sample test I've provided.

### Sample Test

1. What is the default security realm?

- a) LDAP realm
- b) File realm
- c) NT realm
- d) UNIX realm

2. Which configuration file holds the security roles for a Web application?

- a) config.xml
- b) weblogic.xml
- c) web.xml
- d) filerealm.properties

3. When you boot up a WebLogic Server without designating whether the server should be an administrative or managed server, which does it start with?

- a) Administrative
- b) Managed
- c) Neither, you must designate the mode before you start WebLogic server.
- d) None of the above

4. Which configuration file holds the configuration properties for a domain?

- a) config.xml
- b) weblogic.xml
- c) web.xml
- d) filerealm.properties

5. JMX is a:

- a) Performance specification
- b) Messaging specification
- c) Naming and directory specification
- d) System management specification

6. What information do Configuration MBeans provide?

- a) Static server configuration
- b) Runtime resource statistics
- c) True server configuration, based on any overrides
- d) All of the above

7. What does the socketReaders parameter in the administrative console do?

- a) Returns the login timeout for socket readers
- b) Determines whether or not native I/O should be used for the socket readers
- c) Sets the percentage of threads to be available as socket readers
- d) Determines whether tunneling will be enabled for this server

8. Which method of the Pool interface allows

you to shut down the pool and wait for connections to be returned before closing?

- a) ShutdownSoft
- b) Shrink
- c) Reset
- d) DisableDroppingUsers

9. Which load-balancing algorithm has no real guarantee the load will be balanced?

- a) Round-Robin
- b) Random
- c) Weighted Round-Robin
- d) Parameter-based

10. Which interface do you extend to create a message-driven EJB's home interface?

- a) EJBRemote
- b) EJBHome
- c) BeanRemote
- d) None of the above

11. Which configuration file contains the JNDI name of WebLogic-deployed EJBs?

- a) web.xml
- b) config.xml
- c) weblogic-ejb-jar.xml
- d) ejb-jar.xml

12. Which JSP tag includes a JSP page for processing at compile time?

- a) <%@ include file="somepage.jsp" %>
- b) <jsp:include page="somepage.jsp">
- c) <jsp:forward page="somepage.jsp">
- d) None of the above

13. Which Web application directory houses the deployment descriptor file web.xml?

- a) The Web application root
- b) META-INF
- c) WEB-INF
- d) There is no deployment descriptor file named weblogic.xml.

14. Which interface would you most likely use to retrieve properties of an entity bean's EJB container?

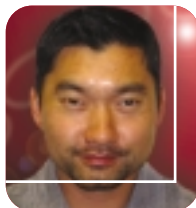
- a) EntityContext
- b) SessionContext
- c) EJBContext
- d) EJBContainer

15. Which of the following is not a valid JSP directive?

- a) page
- b) forward
- c) include
- d) taglib

1. b, 2. c, 3. a, 4. a, 5. d, 6. c, 7. c, 8. a, 9. b, 10. d, 11. c, 12. a, 13. c, 14. a, 15. b





BY Sihyung Park & Oliver Schmelzle

# Completing the J2EE Enterprise Nervous System

JMX AND THE J2EE MANAGEMENT SPECIFICATION

Improving application integration has become an increasingly important component of today's IT strategy. In a recent Morgan Stanley survey of 225 CIOs, 80% indicated that they would begin new application projects in 2002, with application integration as a primary initiative. Why does integration merit top billing? As companies experience escalating economic pressure, application integration enables them to mine additional productivity from existing applications, legacy systems, and new Web services while ensuring that they all work together like a well-oiled machine.

Gartner connects the momentum generated by this application integration movement to the rapid evolution of the established enterprise network. Enterprise networks have grown from simple communication vehicles into highly developed, multi-function Enterprise Nervous Systems (ENS). The ENS, with its role as master mediator of data and applications, is becoming integral to the efficient functioning of a company's key business processes. Although the challenge of constructing an effective ENS is by no means trivial, companies who succeed in

this stand to reap substantial bottom-line benefits, such as agile business processes and rapid, broader dissemination of knowledge throughout different arms of the organization. Today, J2EE technologies represent an ideal technical avenue for constructing and implementing an ENS.

Given the ENS's central role in conducting business processes, companies also must consider the management challenges involved in ensuring that the ENS is available on a 24x7x365 basis after it is deployed. Anything less is unacceptable for a system so critical to a business. As early adopters are discovering, managing the sophisticated technology involved in an ENS is exponentially more challenging than managing the traditional infrastructures for which enterprise management systems were built. Adding to the challenge, the J2EE specification today lacks the functionality to manage distributed environments such as a highly complex ENS. However, effective management can be achieved using a combination of Java Management Extensions (JMX) and the new J2EE Management Specification, previously known as JSR-77.

Among application developers, JMX is gaining in popularity as a standard way to expose business information that enables application-level management. Completing the equation, the J2EE Management Specification addresses the distributed components of a J2EE environment. It provides a concrete object and relationship model of all well-known J2EE components. Every implementation of this specification has to offer a Management EJB (MEJB), which acts as an interface to all managed objects. The MEJB also provides access to state information, event handling, and performance data. JMX and the J2EE Management Specification provide the framework to manage an ENS built on J2EE.

With the hooks established, an ENS management solution must successfully monitor and analyze the thousands of datapoints available to it via JMX and the new J2EE Management Model. Most of the solutions available today are not engineered to handle this high volume of datapoints from multiple systems and technologies distributed throughout the environment. Traditional enterprise management tools

are silo based, requiring IT staff to manually sort through, correlate, and analyze the plethora of data generated across the different servers and subcomponents of the ENS. Given the technical complexity and the business-critical nature of the ENS, this management model simply falls short.

The ENS demands a new way of managing complex, distributed environments – one that is specifically tuned for J2EE architectures. An ENS management solution must be able to automate in real time the correlation of thousands of datapoints based on the complex interdependencies between systems, completely eliminating manual processes on the part of the IT staff. The solution also must be able to perform in-depth analysis on this correlated data to flawlessly isolate faults and suggest resolutions. Even better, the management software should ideally invoke fixes before problems affect the business process.

In essence, what is needed is a "brain" to make the ENS complete: an intelligent and active management approach that would significantly reduce the difficulty of managing these infrastructures and ensure that business processes continue unabated. This would allow companies to not only achieve maximum benefits from their ENS, but also enable them to better leverage their human IT resources. These requirements indeed describe a "next-generation" applications management solution, and it does exist today. This type of solution also lays the groundwork for business process management (BPM), which has yet to be brought to fruition.

If your company is considering implementing an ENS using J2EE technologies, or any other large *n*-tiered applications system, be sure to consider including a "next-generation" management system in your plans. Without it, your nervous system is incomplete.

*continued from page 5*

## Managing Complexity of J2EE

Another example might be automatically configuring J2EE resources rather than requiring a system administrator or developer to maintain this skill set or knowledge base about the data center. For instance, the Evolution Hosting platform ([www.evolutionhosting.com](http://www.evolutionhosting.com)) can automatically configure JDBC connections and mail sessions so that they connect correctly without requiring you to have knowledge of server configuration files (e.g., config.xml) or the data center environment, such as ports or machines and their functions.

Together, development, system administration, and monitoring tools, along with an automated J2EE data center platform, can help manage even the largest production installations. The complexities of J2EE applications will fluctuate with requirements in scalability, security, and flexibility. At least with WebLogic as your platform you are equipped with state-of-the-art development and administration tools, and a slew of vendors whose innovative products fill the gaps.

### AUTHOR BIOS...

Sihyung Park is a senior solutions architect with Covasoftware, Inc. ([www.covasoftware.com](http://www.covasoftware.com)), based in Austin, TX. He has extensive experience as an e-commerce architect in designing and implementing enterprise solutions. Park is the author of *Enterprise Java Beans Using WebLogic 6.1* and has written and taught for the Middleware Company (TheServerSide.com).

Oliver Schmelzle is a sales engineer at Covasoftware, Inc. He has been involved with the development and research of distributed, large-scale applications at international companies such as Sun Microsystems, Bertelsmann, Daimler-Chrysler, and Vignette.

CONTACT: [spark@covasoftware.com](mailto:spark@covasoftware.com)  
[oschmelzle@covasoftware.com](mailto:oschmelzle@covasoftware.com)

# THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany collaboration on a global scale

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

**SPECIAL OFFER!**  
**SAVE \$31\***  
OFFER EXPIRES NOV 30, 2002

Now in More Than 5,000 Bookstores Worldwide – Subscribe **NOW!**

Go Online & Subscribe Today!

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



**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.

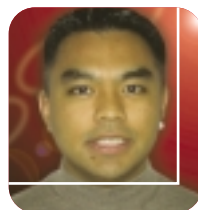


This series of articles focuses on demystifying the frameworks embedded in the BEA WebLogic Portal. The first framework that will be discussed in-depth is the Advisor Framework, followed in future articles by the Event Framework and the Portal Framework.

# Understanding the Advisor Framework

## DEMYSTIFYING THE BEA WEBLOGIC PORTAL FRAMEWORK

BY DWIGHT MAMANTEO



**AUTHOR BIO...**

Dwight Mamanteo is a technical manager with the Global Alliances Technical Services organization at BEA Systems. He has been with BEA since 1999; his current responsibilities include providing advisory and technical enablement support to BEA's strategic SI and ISV partners. He has been involved with object-oriented programming, design, and architecture since 1993.

**CONTACT...**

dwight@bea.com

The Advisor Framework is designed to be an adaptable engine with plug-in points for extending WebLogic Portal's personalization functionality. The main components in the Advisor Framework are the Advisor, AdvisletRegistry, and Advislet classes; the JSP tags; and the Advislet XML Registry. This article will discuss each component, take you through the framework execution scenario, and explain how to implement and configure a custom Advislet.

### Advisor Framework

The key Java components in the Advisor Framework are the Advisor stateless session EJB, the AdvisletRegistry static class, and the different types of Advislet objects (see Figures 1 and 2). At a high level, the personalization JSP tags or a fat Java client would invoke the Advisor with a personalization request; the Advisor would then request the appropriate Advislet from the AdvisletRegistry. After receiving the Advislet, the Advisor would validate the minimal existence and correctness of the request information to ensure that the required data is available to the Advislet. The knowledge concerning what data the Advislet requires is contained within the Advislet itself and is requested by the Advisor during the data validation check. After validating that sufficient data is available, the Advisor requests that the Advislet perform its specific personalization task and return an Advice object that contains the results of the processing. To accomplish the appropriate functionality, the executing Advislet would make use of the different portal services such as the Rules Manager, User Profile Management, and Content Manager.

### ADVISOR

The Advisor is a stateless session EJB that implements the Session Facade J2EE pattern. The Advisor's main responsibility is to encapsulate the business processing required to implement the behavior requested by the personalization JSP tag or fat Java client. As shown in Figure 2, the Advisor gets the Advislet from the AdvisletRegistry, validates the minimal existence and correctness of the request data, and executes the Advislet. After successful completion of the process, an Advice object is returned to the calling client.

### ADVISLETREGISTRY

The AdvisletRegistry is a static class that contains the knowledge of how to parse the URI string being sent by the client and which AdvisletChainElement type to retrieve from its collection. The AdvisletRegistry class also performs the registering and unregistering of the Advislet and AdviceTransform types specified in the "advislet-registry.xml" file into its internal store. The registry pattern used here provides the flexibility of modifying personalization behavior assigned to specific "tags" without requiring the redevelopment of existing code.

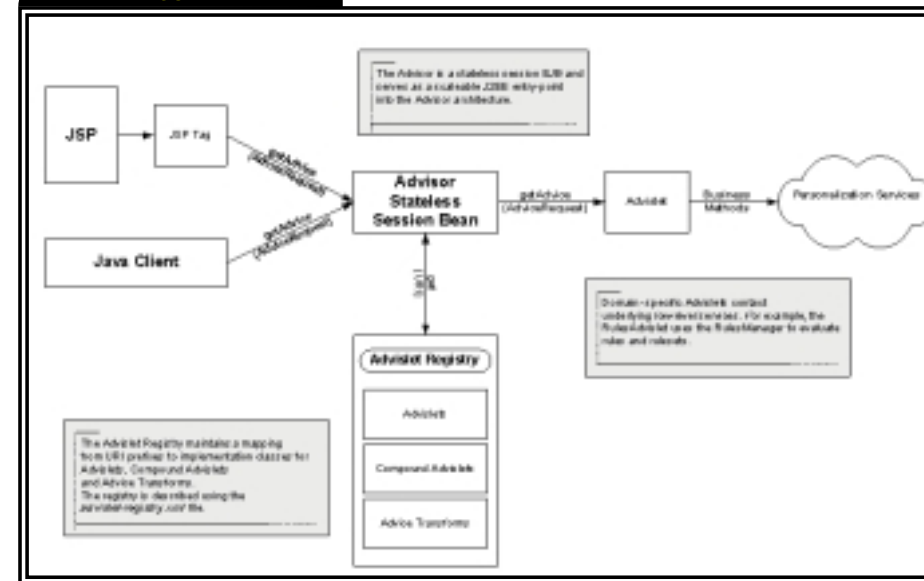
### ADVISLET

Advislets are command objects responsible for interacting with the appropriate services to perform the required personalization task. Out of the box, there are Advislets to classify users, perform documentation searches, and do rules-based matching of user profile information and tagged content. Custom Advislets can also be written to perform specific value-added actions. To write a custom Advislet, you have to implement the Advislet interface and provide implementations for the getAdvice, getRequiredAttributes, and validateAdviceRequest methods (see Figure 3). An abstract class, AbstractAdvislet, is provided to help in the development of custom Advislets. To use the AbstractAdvislet class, simply extend the class, override the getAdvice method, and implement a constructor that takes both the Advisor and metadata objects as arguments. The Advisor argument is a reference to the Advisor creating this Advislet, and the metadata argument is a reference to an object containing the Advislet's name, description, and version information as specified in the Advislet XML Registry.

### COMPOUNDADVISLET

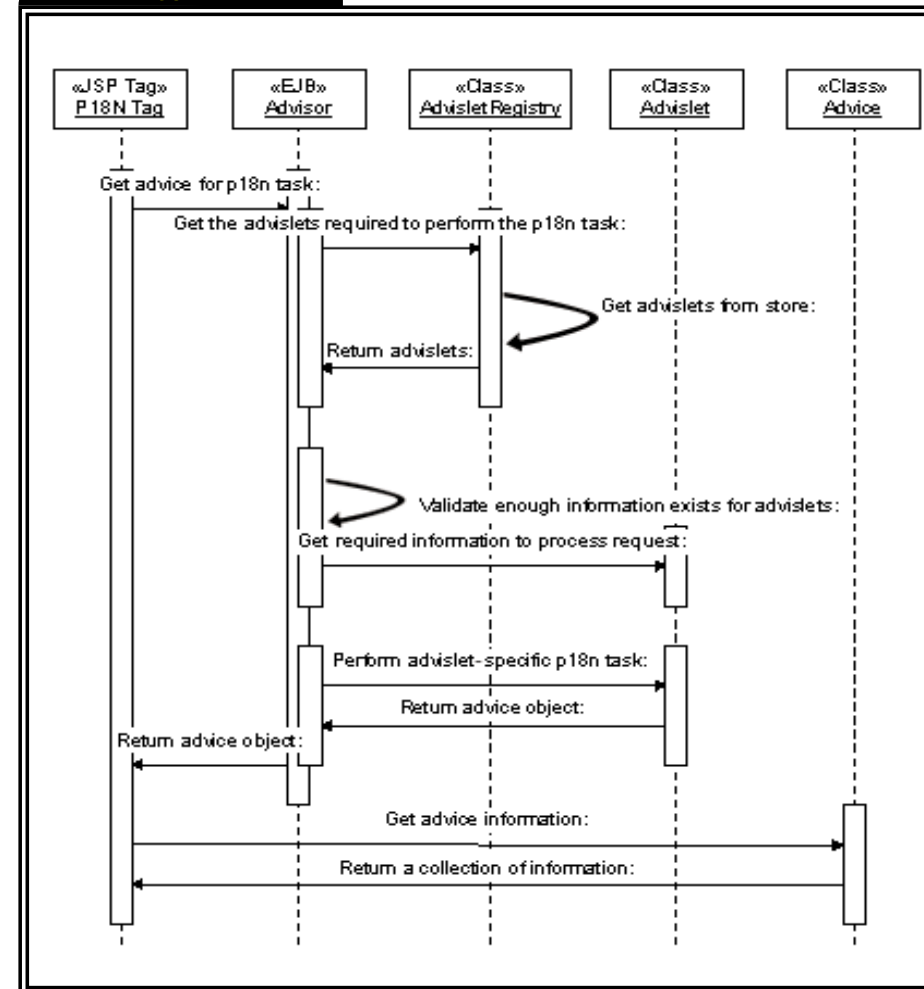
Although the CompoundAdvislet interface is shown in Figure 3, you shouldn't implement this interface or extend the provided CompoundAdvislet

FIGURE 1



High-level Advisor Framework architecture

FIGURE 2



Advisor Framework sequence diagram

implementation. This aspect of the Advislet hierarchy is included simply to inform you that Advislets and AdviceTransforms can be chained together to form a sequence of inter-linking components. If executed, the CompoundAdvislet implementation would request the AdvisletRegistry for each Advislet and AdviceTransform, in the sequence specified in the "advislet-registry.xml" file. The pipe-and-filter pattern used here allows you to develop Advislet and AdviceTransform classes that are independent of each other and link them together to provide a complex personalization process. Additionally, this pattern increases the potential for reuse and adaptability.

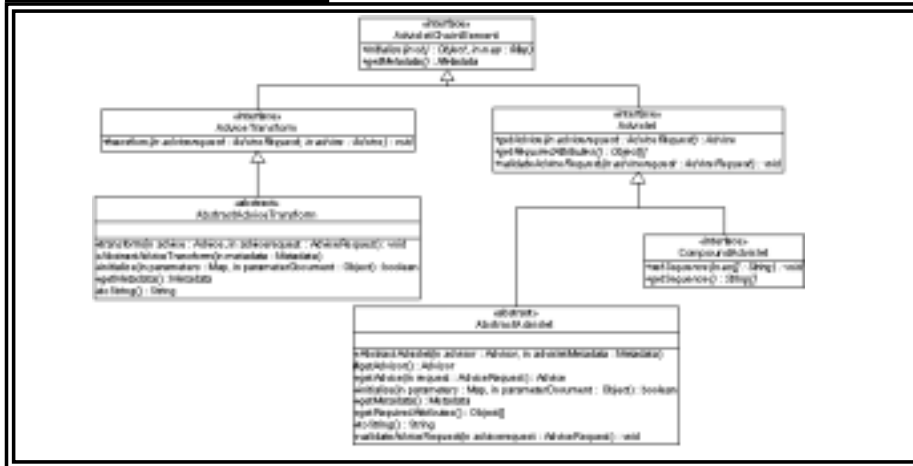
### ADVICETRANSFORM

AdviceTransforms are data-mapping objects responsible for mapping the output results from one Advislet to the input parameters for the next Advislet in the sequence. To write a custom AdviceTransform, a developer would have to implement the AdviceTransform interface and provide an implementation for the transform method (see Figure 3). An abstract class, AbstractAdviceTransform, is provided to help in the development of custom AdviceTransform implementations. To use the AbstractAdviceTransform class, simply extend the class, override the transform method, and implement a constructor that takes both the Advisor and metadata objects as arguments. The Advisor argument is a reference to the Advisor creating this AdviceTransform, and the metadata argument is a reference to an object containing the AdviceTransform's name, description, and version information as specified in the Advislet XML Registry. Out of the box, there are AdviceTransform types to transform standard objects into input objects for the rules engine, the output of the rules engine into the input object for content query, and the results of a classification into the input object for content selection.

### Advislet XML Registry

The Advislet XML Registry contains a listing of all the required Advislet and AdviceTransform elements, their descriptive and configuration information, and the sequence logic for CompoundAdvislets. The information contained in the Advislet XML Registry file is the same information that's loaded in the local store of the AdvisletRegistry class and is scoped to the containing application. To register a custom Advislet in the XML file, simply open the "advislet-registry.xml" file found in the "ejbadvisor.jar" file and add the appropriate

FIGURE 3



Advislet static class diagram

ate lines, making sure to follow the XML Schema provided. The example shown in Listing 1 adds a custom Advislet called MyAdvislet to the registry file.

To register a custom CompoundAdvislet in the XML file, open the same "advislet-registry.xml" file and add the appropriate lines, again making sure to follow the XML Schema provided. The example shown in Listing 2 first adds a custom AdviceTransform called MyAdviceTransform before adding a CompoundAdvislet called MyCompoundAdvislet to the registry file. Notice that the sequence elements of the CompoundAdvislet group list the Advislet and AdviceTransform classes to call and the sequence in which to execute them.

### JSP Tag Library

Three JSP tags are pertinent to the Advisor Framework: <pz: div>, <pz:contentQuery>, and <pz:contentSelector>. These tags allow JSP developers to access the personalization features of WebLogic Portal without having to write EJB client code. (For more information on the parameters for each JSP tag and for the entire WebLogic Portal JSP tag library, visit the BEA online documentation Web site: <http://edocs.bea.com>.)

#### <PZ:DIV>

The <pz:div> JSP tag provides the functionality to wrap JSP code segments within a busi-

ness rule that is dependent upon profile information. That is, a business rule can be called to allow the execution of a JSP code segment within a JSP file only when the profile information matches the business rule requirements. In the example below, the JSP contents contained within the opening (i.e., <pz:div>) and closing (i.e., </pz:div>) tags will be executed only if the rule evaluates to TRUE.

```
<%= taglibs URI="pz.tld" prefix="pz" %>
.
.
.
<pz:div rule="JavaDeveloper">
  <p>This section will be displayed if the
  user's profile has developer=java </p>
</pz:div>
```

This personalization feature allows the runtime dynamic determination of static content and functionality to be displayed to the user.

#### <PZ:CONTENTQUERY>

The <pz:contentQuery> JSP tag provides the functionality to search for tagged content stored in the third-party content management system, which has been integrated into WebLogic Portal using a published content management system SPI. In the example shown in Listing 3, a result set of content objects will be returned that contains

information matching the query string "title='WebLogic Developers Journal'"

This feature provides JSP developers with the ability to dynamically retrieve content contained within the integrated content management system.

#### <PZ:CONTENTSELECTOR>

The <pz:contentSelector> JSP tag combines the functionality of the <pz:div> JSP tag with a content query rule. That is, two actions will occur in sequence when this JSP tag is used, where the second action is dependent upon the first action evaluating as TRUE. In the first action, the Advisor Framework will execute a classification rule that will determine if the profile requirements are met. In the second action, the Advisor Framework will execute a content query rule that will return a result set of content objects that match the query string contained within the rule. In the example shown in Listing 4, a result set of content objects will be returned if the personalization rule evaluates that the profile requirements have been met and that there is tagged content in the content management system that matches the rule's content query string.

This feature provides the ability to decouple the personalized content query logic from the JSP UI development layer. JSP developers can focus on developing UI logic, while the underlying Advisor Framework can focus on performing the required content personalization aspects.

### Conclusion

The Advisor Framework enables the creation of an application that allows business rules to drive the presentation of content and functionality to the right person at the right time. The out-of-the-box Advisor Framework components include personalization JSP tags, a runtime engine, an XML Advislet configuration file, and components that classify users, enable documentation searches, and perform rules-based matching of user profile information and tagged content. In addition, the framework also provides an extendable interface that allows the easy inclusion of custom personalization components. 🍷

#### Listing 1

```
<advislet>
<registration-key>SampleAdvislet</registration-key>
<metadata>
<name>MyAdvislet</name>
<description>A sample Advislet that prints a message to the
  console.</description>
<author>Dwight Mamanteo</author>
```

```
<version>
<description>WLP 7.0</description>
<major>7</major>
<minor>00</minor>
<build-number>1</build-number>
</version>
</metadata>
<implementation class>WLDJ.sample.advislets.MyAdvisletImpl</imple-
```

```
metation-class>
</advislet>

Listing 2
<advice-transform>
<registration-key>SampleTransform</registration-key>
<metadata>
<name>MyTransform</name>
<description>A sample AdviceTransform that prints a message to
  the console.</description>
<author>Dwight Mamanteo</author>
<version>
<description>WLP 7.0</description>
<major>7</major>
<minor>00</minor>
<build-number>1</build-number>
</version>
</metadata>
<implementationclass>WLDJ.sample.advislets.MyAdviceTransformImpl
</implementation-class>
</advice-transform>

<compound-advislet>
<registration-key>SampleCompoundAdvislet</registration-key>
<metadata>
<name>MyCompoundAdvislet</name>
<description>A sample CompoundAdvislet that prints two messages
  to the console.</description>
<author>Dwight Mamanteo</author>
<version>
<description>WLP 7.0</description>
<major>7</major>
<minor>00</minor>
<build-number>1</build-number>
</version>
</metadata>
<sequence>
<advice-transform>SampleTransform</advice-transform>
<advislet>SampleAdvislet</advislet>
```

```
</sequence>
</compound-advislet>

Listing 3
<%= page import="com.bea.pl3n.content.ContentHelper"%>
<%= taglib URI="pz.tld" prefix="pz" %>
.
.
.
<pz:contentQuery id="journals" contentHome="<%=ContentHelper.DEF_DOCU-
  MENT_MANAGER_HOME %>" query="title = 'WebLogic Developers Journal' " />
<ul>
<es:forEachInArray array="<%=journals%>" id="aJournalEdition"
  type="com.bea.pl3n.content.Content">
<li>The Main Article is: <cm:printProperty id="aJournalEdition"
  name="MainArticle" encode="html" />
<li>The Journal Edition is: <cm:printProperty id="aJournalEdition"
  name="Edition" encode="html" />
</es:forEachInArray>
</ul>

Listing 4
<%= page import="com.bea.pl3n.content.ContentHelper" %>
<%= taglib URI="cm.tld" prefix="cm" %>
<%= taglib URI="pz.tld" prefix="pz" %>
<%= taglib URI="es.tld" prefix="es" %>
.
.
.
<pz:contentSelector id="journals" rule="JournalForJavaDevelopers"
  contentHome="<%=ContentHelper.DEF_DOCUMENT_MANAGER_HOME %>" />
<ul>
<es:forEachInArray array="<%=journals%>" id="aJournal"
  type="com.bea.pl3n.content.Content">
<li>The Main Article is: <cm:printProperty id="aJournalEdition"
  name="MainArticle" encode="html" />
<li>The Journal Edition is: <cm:printProperty id="aJournalEdition"
  name="Edition" encode="html" />
</es:forEachInArray>
</ul>
```

# Performant

[www.performant.com/weblogic1](http://www.performant.com/weblogic1)



Security is a priority for most of our customers. As more and more customers adopt Web services, they find a need to understand how Web services can be secured and what authentication mechanism to use. In order to keep Web services open and support multiple client types, it's necessary to understand how to handle Web-services security.

the following specific components:

- A Web application that contains, at a minimum, a servlet that sends and receives SOAP messages to and from the client. It's automatically included as part of the Web services development process.
- A stateless session EJB that implements an RPC-style Web service or a JMS listener (such as a message-driven bean) for a message-style Web service.

In an RPC-style Web service, the stateless session EJBs might do all the actual work of the Web service, or they may parcel out the work to other EJBs. The implementer of the Web service decides which EJBs do the real work. In a message-style Web service, a J2EE object (typically a message-driven bean) gets the messages from the JMS destination and processes them.

WebLogic Web services are packaged as Enterprise archive (.ear) files that contain the Web application's Web archive (.war) files and EJB archive (.jar) files.

## Securing WebLogic Web Services

Since WebLogic Web services are packaged as standard J2EE Enterprise applications, access to a Web service may be secured by securing some or all of the following standard J2EE components that make up the Web service:

- The SOAP servlets
- The stateless session EJB upon which an RPC-style Web service is based

Basic HTTP authentication or SSL can be used to authenticate a client that is attempting to access a WebLogic Web service. Because the preceding components are standard J2EE components, they may be secured using standard J2EE security procedures.

## SECURING MESSAGE-STYLE WEB SERVICES

A message-style Web service may be secured by securing the SOAP servlet that handles SOAP messages between the client and the service.

When the Web service is assembled, either manually or by using the wsgen Ant task, you reference SOAP servlets in the web.xml file of the Web application. These servlets handle the SOAP messages passed between WebLogic Server and client applications. They are always deployed on WebLogic Server and are shared by all deployed WebLogic Web services.

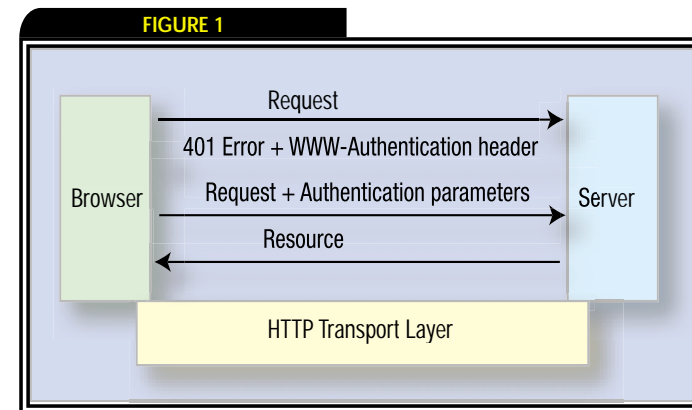


FIGURE 1  
HTTP 1.1 Challenge Response Authentication

The particular SOAP servlet referenced by a Web service depends on its type (RPC-style or message-style). The following list describes each SOAP servlet:

- **weblogic.soap.server.servlet.DestinationSendAdapter:** Handles SOAP messages in a message-style Web service that receives data from a client application and sends it to a JMS destination
- **weblogic.soap.server.servlet.QueueReceiveAdapter:** Handles SOAP messages in a message-style Web service that sends data from a JMS Queue to a client application
- **weblogic.soap.server.servlet.TopicReceiveAdapter:** Handles SOAP messages in a message-style Web service that sends data from a JMS Topic to a client application
- **weblogic.soap.server.servlet.StatelessBeanAdapter:** Handles SOAP messages between an RPC-style Web service and a client application

For example, in a message-style Web service in which client applications send data to a JMS destination, the SOAP servlet that handles the SOAP messages is weblogic.soap.server.servlet.DestinationSendAdapter. The wsgen Ant task used to assemble the Web service adds the elements shown in Listing 1 to the web.xml deployment descriptor of the Web application.

To restrict access to the DestinationSendAdapter SOAP servlet, you first define a role that is mapped to one or more principals in a security realm, then specify that the security constraint applies to this SOAP servlet by adding the following url-pattern element inside the web-resources-collection element to the web.xml deployment descriptor of the Web application:

```
<url-pattern>/sendMsg</url-pattern>
```

## SECURING AN RPC-STYLE WEB SERVICE

You can restrict access to an RPC-style Web service by restricting access to the stateless session EJB that implements the Web service or the SOAP servlet.

Listings 2 and 3 show how to set up the web.xml and weblogic.xml files to secure the SOAP servlets. Make sure that the WSDL file is still accessible by the clients. To ensure that, avoid using wild-card mapping; instead secure the SOAP servlet adapter path explicitly. The listings provide an example of how to secure the account manager proxy Web service. As you can see, the role is defined in web.xml and is associated to the group in the realm.

## WEB SERVICES CLIENTS

In order to understand how security for Web services clients is handled, it's necessary to understand how SOAP and HTTP authentication works.

## SOAP Authentication

Web services use the SOAP protocol, a high-level messaging protocol implemented over some underlying transport mechanism. Web services security is still an emerging field and there are extensions emerging around the SOAP specification to support security features. Currently, SOAP uses the underlying transport protocol infrastructure for authentication. So WebLogic Server SOAP indirectly uses HTTP 1.1 authentication.

## HTTP Authentication

The method for user authentication used with HTTP is quite simple. Since HTTP is a stateless protocol – that is, the server doesn't remember any information about a request once it has finished – the browser needs to resend the username and password on each request (see Figure 1).

On the first access to an authenticated resource, the server will return a 401 status (“Unauthorized”) and include a WWW-Authenticate response header, which will indicate the realm name and which authentication scheme to use. The browser should then ask the user to enter a username and password. It then requests the same resource again, this time including an authorization header that contains the scheme name (“Basic”) and the username and password entered.

The server checks the username and password, and if they are valid, returns the page. If the password is not valid for that user, or the user is not allowed access, the server returns a 401 status as before. The browser can then ask the user to retry the username and password.

Assuming the username and password are valid, the user might next request a protected resource. In this case, the server would respond with a 401 status, and the browser could send the request again with the user and password details. This would be slow, however, so instead the browser sends the authorization header on subsequent requests. Refer to the W3C HTTP Working Group RFC 2617 ([www.w3c.org/Protocols/Specs.html](http://www.w3c.org/Protocols/Specs.html)) for more details on HTTP authentication.

## CLIENT TYPES

Even though the previous section focused on browser clients, the protocol is the same for any type of client. Any client that needs to access secured Web services is expected to understand this HTTP authentication protocol and implement it in order to be authenticated. This section shows how authentication information can be passed from various types of clients.

## MS Visual Basic Clients

Microsoft provides an HTTPConnector interface for the transport layer. It uses two properties, AuthUser and AuthPassword, to pass credentials. Listing 4 is an example of invoking a secured WebLogic Web service using Visual Basic and MS SOAP Toolkit 2.0.

AuthUser and AuthPassword should not be confused with ProxyUser and ProxyPassword. They are used to provide the credentials for the proxy server, if there are any. This authentication

# WebLogic Web Services Security

A LOOK UNDER THE HOOD

BY ANBARASU KRISHNASWAMY



This article takes a look under the hood of WebLogic Web services security. I'll explain how WebLogic Web services can be secured, how authentication works, and how to develop clients in various programming languages to authenticate against WebLogic Web services.

## WebLogic Web Service Components

Web services hosted by WebLogic Server are implemented using standard J2EE components such as EJBs and JMS, and are packaged as standard J2EE enterprise applications. WebLogic Web services use Simple Object Access Protocol (SOAP) 1.1 as the message format and HTTP 1.1 as the connection protocol.

The Web services runtime component is a set of servlets and associated infrastructure needed to create a Web service. One element of the runtime is a set of servlets that handle SOAP requests from a client. These servlets are included in the WebLogic Server distribution. Another element of the runtime is an Ant task that generates and assembles all the components of a WebLogic Web service.

WebLogic Web services are packaged as standard J2EE Enterprise applications that consist of

## AUTHOR BIO...

Anbarasu Krishnaswamy, a senior principal consultant with BEA Professional Services, has several years of experience with BEA products and is a Sun-certified Java programmer and BEA-certified WebLogic Server developer. Anbarasu holds a master's degree in computer science and engineering and a bachelor's degree in electrical and electronics engineering.

## CONTACT...

anbarasu@bea.com

works the same way, except that an error number 407 is returned instead of 401.

#### Java Clients

WebLogic Server provides Java client libraries to access secured resources. Usernames and credentials are passed using "java.naming.security.principal" and "java.naming.security.credentials". Listing 5 describes how a Java client would invoke a secured WebLogic Web service using WebServiceProxy.

#### JAX-RPC Clients

JAX-RPC (Java API for XML-based remote procedure calls) is a new standard released in June 2002 that defines APIs for invoking Web services. WebLogic Server 7.0 supports JAX-RPC. The JAX-RPC interfaces "Stub" and "Call" both support the properties "javax.xml.rpc.security.auth.username" and "javax.xml.rpc.security.auth.password" with which the authentication information may be passed. You can also use the static constants Stub.USERNAME\_PROPERTY and Stub.PASSWORD\_PROPERTY to set the security information before making a service call (see Listing 6). The username and password may also be passed while getting the port using get<web service>port().

#### MS C++ Clients

Similarly, you can write a C/C++ client using APIs provided by the vendor. The MS SOAP toolkit may be used to write C++ clients on a Windows platform. The username and password are passed using Connector Property. Following is an example that sets the username and password.

```
Connector->Property["AuthUser"] = "<UserName>";
Connector->Property["AuthPassword"] = "<Password>";
```

#### Conclusion

This article discussed authentication for WebLogic Web services and securing Web services components. As Web services are a natural choice for heterogeneous environments involving different technologies, it's necessary to understand how to access secured Web services using clients of various types. 🍷

#### Listing 1

```
<servlet>
  <servlet-name>sender</servlet-name>
  <servlet-class>
weblogic.soap.server.servlet.DestinationSendAdapter
  </servlet-class>
  <init-param>
    <param-name>topic-resource-ref</param-name>
    <param-value>senderDestination</param-value>
  </init-param>
  <init-param>
    <param-name>connection-factory-resource-ref</param-name>
    <param-value>senderFactory</param-value>
  </init-param>
</servlet>
...
```

```
<servlet-mapping>
  <servlet-name>sender</servlet-name>
  <url-pattern>/sendMsg</url-pattern>
</servlet-mapping>
```

#### Listing 2

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name> Weather</web-resource-name>
    <url-pattern>/weatherturi/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>Admin</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>
<security-role>
  <role-name>Admin</role-name>
</security-role>
```

#### Listing 3

```
<security-role-assignment>
  <role-name>Admin</role-name>
  <principal-name>system</principal-name>
</security-role-assignment>
```

#### Listing 4

```
Set Client = New SoapClient
Client.mssoapinit "<WSDL URL>", "<Service Name in WSDL>", "<Port Name in WSDL>"
Connect
Client.ConnectorProperty("AuthUser") = <username>
Client.ConnectorProperty("AuthPassword") = <password>
Client.<Service(args)>
```

#### Listing 5

```
Properties h = new Properties();
h.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.soap.http.SoapInitialContextFactory");
h.put("java.naming.security.principal", "<username>");
h.put("java.naming.security.credentials", "<password>");
Context context = new InitialContext(h);
WebServiceProxy proxy = (WebServiceProxy)context.lookup("<WSDL_URL>");
SoapMethod method = proxy.getMethod("<method name>");
method.invoke(new Object[]{"<args>"});
```

#### Listing 6

```
<Web Service Stub>._setProperty(Stub.USERNAME_PROPERTY, "<username>");
<Web Service Stub>._setProperty(Stub.PASSWORD_PROPERTY, "<password>");
```

# BEA Systems

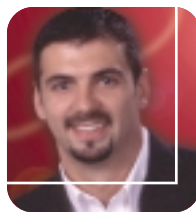
<http://dev2dev.bea.com/useworkshop>



# The Evolution Continues

MONITORING WEB APPLICATIONS MOVES INTO TRANSACTIONS AND WEB SERVICES

BY FRANK MORENO



## AUTHOR BIO...

Frank Moreno is the product marketing manager for Dirig Software, a leading developer of award-winning enterprise performance management solutions. Frank has over 10 years of experience in product marketing, product management, and strategic alliances in the networking and software industries, and has written multiple articles on e-business performance management.

## CONTACT...

fmoreno@dirig.com

As developers rapidly embraced the use of component-based architectures, the role of application servers in production has expanded from hosting somewhat simple, servlet-based applications to exploiting Enterprise JavaBeans (EJBs) and Java messaging services (JMS) to build robust eBusiness applications.

The proliferation of new, online business applications has sparked technology innovations in what was once a completely separate industry – performance management solutions. In the past 12–18 months, two previously unrelated areas, development and IT operations, have been forced to work together with new monitoring and management products that can pinpoint performance issues down to an individual Java method. These production-oriented management products take advantage of technologies such as management APIs, instrumentation, and Java profiling, which have been incorporated with more traditional management architectures such as server-side agents. Regardless of the granularity that these tools offer, the management of these increasingly complex *n*-tier infrastructures is still applied in a disparate, stovepipe approach, with multiple departments owning the operations responsible for various layers. This often results in reactive finger-pointing and departmental blame when performance issues arise.

## Beyond App Server Management

In less than two years, a relatively new market niche (performance management of J2EE application servers) has become saturated by a variety of vendors. Each solution is slightly different, but all identify essentially the same data – hundreds of statistics for various application servers and

their components. The opportunity now exists for management software vendors to take the next step and provide both a view into, and management of, the actual application topology. This visibility will extend beyond the application server and address management of the complete business process infrastructure with monitoring of transaction dependencies and the paths available to execute a transaction.

New management solutions are leading the evolution from traditional systems management to what AMR Research refers to as Business Services Management. “Many different pieces of technology are assembled to deliver a business service, including Web servers, application servers, and database servers. Monitoring and managing all of this technology in the context of the business processes it supports is the cornerstone of Business Services Management.”

While yesterday’s tools provide massive amounts of data for the various layers, often resulting in too much information, – known as a “data glut” – IT still struggles to gain a view of the paths each transaction could potentially take throughout this complex architecture. At the same time, business owners are looking to IT for details on the transaction ecosystem to help measure the success or failures of their online initiatives.

Moreover, as more and more companies look to maximize their application server infrastructures by launching new, online business initiatives, IT lacks visibility into the disconnected or distributed applications that many transactions depend upon to execute. There is still a need to track the real-time transaction dependencies to understand what each component requires to execute and how those components behave.

Market research firm Hurwitz Group underscores this growing management requirement, “Hurwitz Group believes that application management folks will have to borrow some network management concepts (such as topology mapping and route tracing) to address this problem. We are looking for a solution that can create a software topology – a map of the various software components in the clustered production environment that can make up transactions. The solution should also be able to trace the particular route that a transaction is taking across the software topology. Only then can administrators match individual transactions with the specific components that were used – the uncertainty disappears and problem diagnosis can begin.”

## Mapping the Application Logic

Today, new technologies are being introduced to monitor the real-time J2EE application logic. Mapping the individual business processes, or “Transaction Path Mapping,” is the next step in Web application management solutions. Correlating component-level performance data with the system, database, application, and business-related metrics allows IT operations to measure the true performance of an e-business system and translate it into relevant, business-related information. This look into the application logic can ultimately bring management functionality more in line with the business and online revenue goals of the organization.

While load balancing, firewalls, and clustering are essential to the scalability and performance of a Web application, they also add complexities that pose new IT obstacles. Transaction Path Mapping identifies the components required to execute an application and creates a topology view of the potential paths available for transactions to take. The cross-functional view incorporates each element or layer that was previously monitored in its own “stovepipe” view, enabling IT to drill down to the exact source of the problem causing a transaction failure.

By correlating the various data points monitored, the overall health of a transaction path can be displayed, allowing for transactions to be routed to a healthier path, thereby maintaining or increasing transaction success rates. Additionally, the transaction map can help isolate the location of a transaction failure within an infrastructure. More importantly, this level of monitoring enables IT operations to communicate to both development and line-of-business managers the detail they require.

## New Concept, Different Approaches

Similar to the various component-monitoring solutions, vendors are developing different approaches to transaction mapping. One popular method requires a manual mapping process to outline an application diagram. Typically, this takes advantage of the developer’s application architecture and requires IT to select the dependencies of each transaction. This methodology is merely a representation of the infrastructure. It fails to monitor the real-time application logic and requires constant maintenance to the dependency map as new components are added or deleted. Other tools will require C or C++ development to actually hard-code the mapping into a model, sacrificing flexibility and requiring resources that are typically outside of IT. While this method may provide an accurate blueprint of the original, intended architecture, the continual changes required to update the

mapping of evolving component-based applications may be costly. Another approach to identifying transaction paths is the use of transaction labels or tags. This method involves the monitoring solution labeling a transaction and tracking the actual path it takes through all layers of the infrastructure. It is ideally suited for pre-production environments and offers detailed component information, down to individual SQL statements and method calls, but requires a significant amount of JVM and byte-code instrumentation. Combined with the added overhead from the label applied to each transaction, it can have a significant adverse effect on performance in a production environment.

The most effective approach to transaction mapping will likely be a combination of a monitoring agent for each system, with the ability to monitor the entire Web stack (Web server, application server, and database) and some form of instrumentation (controlled by IT) or API communication for component-level statistics. By leveraging a “bottom-up” approach with real-time data, not only can IT establish a live view of each transaction path, but the view can be dynamic, to ensure an accurate representation of the infrastructure. Any changes to the topology can be tracked without the need for development time or resources.

Beginning with a transaction entry point at the presentation layer, typically a Web page serving up a servlet or active server page, IT can auto-discover each servlet, EJB, or COM+ object. EJB transactional methods, JDBC connection; or other resource pools that make up the business logic. This information can then be used to dynamically map the real-time flow of each path and represent it visually in a portal-style view.

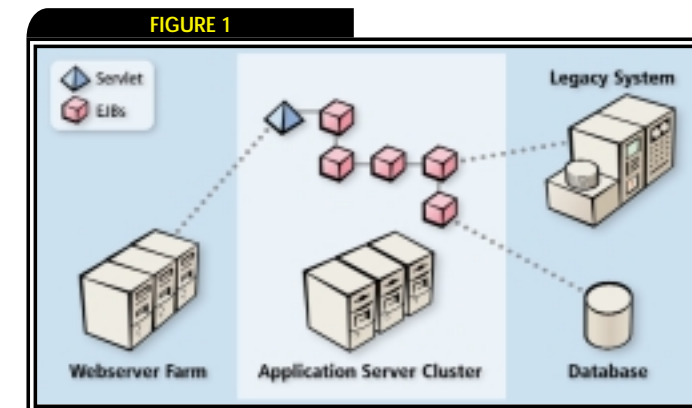
This methodology offers maximum visibility and detail for data correlation while avoiding costly overhead on production systems. It also fits in with the flexibility required to support rapidly evolving Web architectures, which may add, delete, or modify production-side applications and components on a regular basis. More importantly, this approach allows for the mapping solution to invoke various actions automatically, based on defined thresholds. These can be corrective measures to address poor performance, or even quality-of-service actions to prioritize resources based on a particular transaction trait (i.e., a preferred customer or large transaction value).

With IT operations and development forced to collaborate when addressing Web application performance issues, products that address the needs of both groups are more apt to be embraced. Because these solutions can display more detailed performance information in a manner that is familiar to IT, the need to involve development in less complex issues is reduced. The result is significantly increased productivity from both operations and development.

As e-business applications become the foundation for delivering business services online, the ability to measure, monitor, and proactively manage all aspects of the service becomes increasingly important. In fact, the introduction of reliable transaction management technologies will likely help drive the adoption of new online initiatives. This new approach to Web application performance management will set the stage for management technology innovations in the near future.

## References

- Gaughan, Dennis (March 11, 2002). “Business Services Management: Managing an ECM Infrastructure.”
- Hurwitz TrendWatch, March 15, 2002. *Web Application's Uncertainty Principle.*



Transaction mapping provides dependencies



## performance

The concept of persisting a user session during the interaction with an application server has matured from maintaining hidden HTML fields and toying with URLs to a stable and robust technology under the J2EE framework. This is what is commonly referred to as an HTTP session – a conversation that spans multiple requests between a client and the server.

# Session Persistence Performance in BEA WebLogic Server 7.0

FINDING THE SOLUTION THAT BEST MEETS YOUR NEEDS

The session could be maintained at the various layers on the application architecture. It could reside on the client, on the presentation layer, in the object layer, or even in the database layer of the application. There are two options for maintaining client sessions when session data resides on the client layer. Session information can be maintained in hidden HTML fields or, alternatively, via the use of cookies, which are maintained on the client. This strategy has some inherent limitations, such as only String values can be stored and an increase in the size of session data increases the associated network overheads, which is a detriment to obtaining optimal performance. The lack of any persistence mechanism makes this approach vulnerable to client crashes.

Maintaining sessions on the presentation or the Web application layer is a better option from the perspective of performance, reliability, and usability. This strategy mitigates some of the earlier problems associated with client-side session management. It provides additional options to manage the session data, reduces network overheads, and imposes no constraints on the type and size of the objects that can be stored in the session. It also provides failover mechanisms for session state recovery in the case of server or cluster crashes. In addition, this approach utilizes

various application server-specific features (clusters, for instance) that increase the reliability, scalability, and performance of the application.

Session data can also be maintained at the object layer by storing the data in stateful EJBs. This would utilize the Web application layer to store the handles of these stateful EJBs so clients can access the same bean to get the required information. This is a good approach when the application data is predominantly present in the object layer and the Web application layer is used primarily as an interface to the system. This combination results in a lightweight HTTP session and could potentially help improve the performance of the application. However, the overhead as a result of EJB activation and passivation need to be carefully considered.

WebLogic Server 7.0 provides a significant amount of flexibility and choice in determining the appropriate persistence mechanism for a wide variety of application needs. The session management is completely transparent to the user and can be easily configured via the deployment descriptor file of the Web application. This article focuses on the various HTTP session persistence mechanisms and their performance characteristics (in relative terms) for varying sizes of session data. There are five different implementations of session persistence: memory (single-server, nonreplicated), cookie-based, file system persistence, JDBC persistence and in-memory replication (across a cluster). These options, available in WebLogic Server 7.0, are in marked contrast to the competition, which only offers a JDBC persistence mechanism for clustered environments.

For the results shown here, a StringBuffer object is stored in the HTTP session and the size of the session data is varied by increasing the length of the StringBuffer object. The client measures the server response and calculates the throughput for 50 concurrent users. The graphs show the relative performance throughput for various sizes of session data size. In each case, the baseline number represents the throughput obtained when a StringBuffer object of unit length is stored in the session and the throughputs are computed relative to this baseline number.

### Memory-Based Session Persistence

This is the default mechanism for managing sessions. The term "memory" refers to the server (JVM process) memory where the Web application is deployed and executed. The session data is maintained (in the JVM process memory) for the life cycle of the session object. To use the memory-

BY SAURABH DIXIT & SRIKANT SUBRAMANIAM

#### AUTHOR BIOS...

Saurabh Dixit and Srikanth Subramaniam are engineers on the Performance Team at BEA's WebLogic Division

#### CONTACT...

saurabh@bea.com  
srikant@bea.com

\$195  
FOR SYS-CON  
SUBSCRIBERS

BEST EDUCATIONAL VALUE  
OF THE YEAR!

TO REGISTER: [www.sys-con.com](http://www.sys-con.com) or Call 201 802-3069

web services **EDGE**  
world tour 2002

# Take Your Career to the Next Level!

EACH CITY WILL BE SPONSORED BY A LEADING WEB SERVICES COMPANY

SHARPEN YOUR PROFESSIONAL SKILLS.

KEEP UP WITH THE TECHNOLOGY EVOLUTION!

"Presented an excellent overview of Web services. Great inside knowledge of both the new and old protocols. Great insight into the code piece."  
—Rodrigo Frontecilla

"Very articulate on the Web services SOAP topic and well-prepared for many questions. I've learned a lot from this seminar and I appreciate this seminar for my job. Thank you!"  
—Kenneth Unpingco, Southern Wine & Spirits of America

"I liked the overview of Web services and the use of specific tools to display ways to distribute Web services. Good for getting up to speed on the concepts."  
—B. Ashton, Stopjetlag.com

Echoed over and over by Web Services Edge World Tour Attendees:

"Good balance of theory and demonstration."

"Excellent scope and depth for my background at this time. Use of examples was good."

"It was tailored toward my needs as a novice to SOAP Web services – and they explained everything."

WHO SHOULD ATTEND:

- Architects
- Developers
- Programmers
- IS/IT Managers
- C-Level Executives
- i-Technology Professionals

## Learn How to Create, Test and Deploy Enterprise-Class Web Services Applications

TAUGHT BY THE INNOVATORS AND THOUGHT LEADERS IN WEB SERVICES

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

### SPONSOR A CITY!

Position your company as a leader in Web services

Call 201 802.3066 to discuss how

IF YOU MISSED THESE...

BOSTON, MA (Boston Marriott Newton) **SOLD OUT!**  
WASHINGTON, DC (Tysons Corner Marriott) **SOLD OUT!**  
NEW YORK, NY (Doubletree Guest Suites) **SOLD OUT!**  
SAN FRANCISCO, CA (Marriott San Francisco) **SOLD OUT!** **CLASSES ADDED**

BE SURE NOT TO MISS THESE...

...COMING TO A CITY NEAR YOU

2002  
LOS ANGELES ..... SEPTEMBER 19  
SAN JOSE AT WEBSERVICES EDGE 2002... **WEST** ..... OCTOBER 1  
CHICAGO ..... OCTOBER 17  
ATLANTA ..... OCTOBER 29  
MINNEAPOLIS ..... NOVEMBER 7  
NEW YORK ..... NOVEMBER 18  
SAN FRANCISCO ..... DECEMBER 3  
BOSTON ..... DECEMBER 12

2003  
CHARLOTTE ..... JANUARY 7  
MIAMI ..... JANUARY 14  
DALLAS ..... FEBRUARY 4  
BALTIMORE ..... FEBRUARY 20  
BOSTON ..... MARCH 11

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE LOWEST REGISTRATION FEE.

TOPICS HAVE INCLUDED: Developing SOAP Web Services  
Architecting J2EE Web Services

On May 13, 2002, the San Francisco tutorial drew a record 601 registrations.

PRESENT YOUR COMPANY'S EXPERTS TO AN EAGER AUDIENCE  
READY TO LEARN FROM YOU! ACT TODAY!

i-TECHNOLOGY  
SYS-CON EDUCATION

REGISTRATION FOR EACH CITY CLOSES THREE BUSINESS DAYS BEFORE EACH TUTORIAL DATE. DON'T DELAY. SEATING IS LIMITED. NON-SUBSCRIBERS: REGISTER FOR \$245 AND RECEIVE THREE FREE ONE-YEAR SUBSCRIPTIONS TO WEB SERVICES JOURNAL, JAVA DEVELOPER'S JOURNAL, AND XML-JOURNAL, PLUS YOUR CHOICE OF BEA WEBLOGIC DEVELOPER'S JOURNAL OR WEBSERVICES DEVELOPER'S JOURNAL, A \$345 VALUE!

TO REGISTER:  
[www.sys-con.com](http://www.sys-con.com)  
or Call 201 802-3069

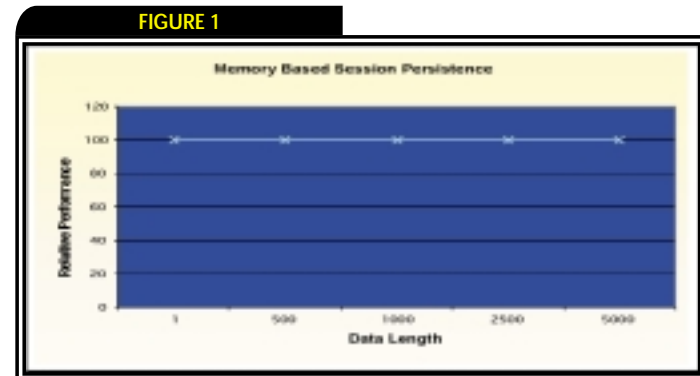


based, single-server, nonreplicated persistent storage, set the PersistentStoreType property in the <session-descriptor> element of the WebLogic-specific deployment descriptor, weblogic.xml to *memory*. When you use memory-based storage, all session information is stored in memory and is therefore lost when you stop and restart WebLogic Server.

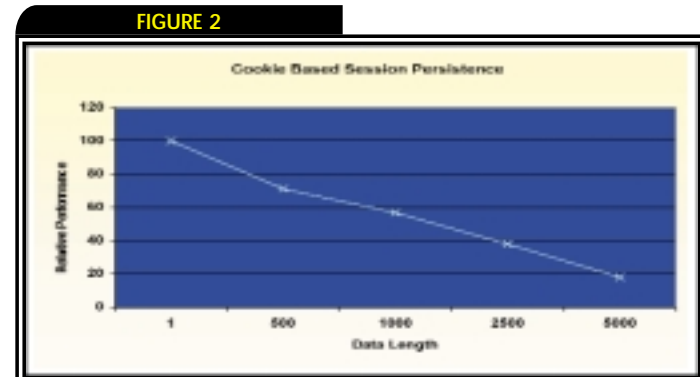
This approach provides very good performance but does not provide any failover capability (due to the lack of any data replication as mentioned above). Benchmark results (see Figure 1) indicate stable performance irrespective of the size of the session data.

### Cookie-Based Session Persistence

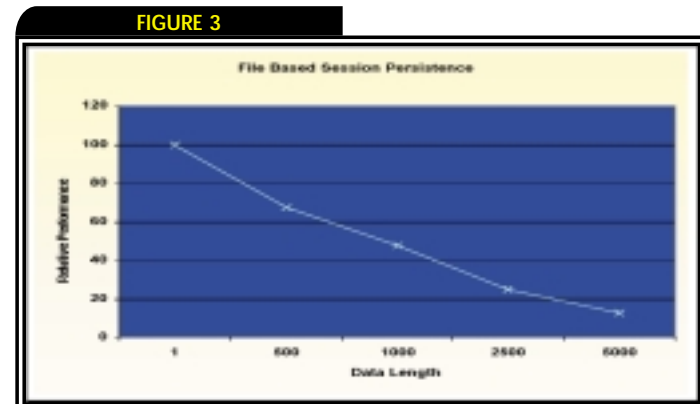
This maintains the session state on the client, which is the biggest difference between this approach and the other mecha-



Memory-based session persistence



Cookie-based session persistence



File-based session persistence

nisms. There are two ways to set up cookie-based session persistence: set the PersistentStoreType property in the <session-descriptor> element of weblogic.xml to *cookie*; or, optionally, set a name for the cookie using the PersistentStoreCookieName parameter. The default is WLCOOKIE.

This approach is most useful when you don't need to store large amounts of data in the session and it doesn't require the use of WebLogic clusters. Since the session is stored on the client (and not on the server), you can start and stop WebLogic Servers without losing sessions. The major limitation of cookie-based persistence is that only string attributes can be stored in the session.

### File-Based Session Persistence

This is a cost-effective solution, as it relies on the underlying file system to persist data and thus eliminates the need for a database. To configure file-based persistent storage for sessions, set the PersistentStoreType property in the <session-descriptor> element of weblogic.xml to *file* and specify the directory where WebLogic Server stores the sessions (defined by the PersistentStoreDir attribute in weblogic.xml). WebLogic Server creates various subdirectories under this directory to store session data in the form of files. The creation of this root directory and all other subdirectories and files used to maintain session information is automatically taken care of by the WebLogic Server and these files are generated, updated, and deleted based on the session life-cycle events. The failover mechanism offered under this approach depends upon how the root directory has been defined. This could be set to the name of a directory or a common directory on the local machine, or it could be a shared directory on the network. Making the root directory a shared directory among the different servers in a cluster makes file-persistent sessions resilient to failures.

It is important to ensure that you have enough disk space to store the number of valid sessions multiplied by the size of each session. The size of a session can be determined by looking at the files created under the root directory. (Note that the size of each session can vary as the size of serialized session data changes.) From a performance perspective, file persistence is the slowest mechanism for managing sessions.

### JDBC-Based Session Persistence

This configuration doesn't require the use of WebLogic clusters and gives users the option of maintaining the session state for longer periods of time (as the session information is persisted in a database). To configure JDBC-based persistent storage for sessions, set the PersistentStoreType property in the <session-descriptor> element of weblogic.xml to *jdbc* and specify a JDBC connection pool to be used for persistence storage with the PersistentStorePool property (additional required configuration steps are described in the references at the end of this article).

The use of a database also offers the most reliable solution for data persistence and failover in the event of server crash. Since session data is stored in a database, any server can access the session through use of a session identifier that serves as the client's session identity in the database.

The choice of the database, JDBC driver, and JDBC connection pool configuration affects the overall performance of this approach. While all session persistence mechanisms have to deal with the overheads of data serialization and deserialization, the additional overhead of the database interaction impacts the performance of the JDBC-based session persistence and causes it to

**Once you're in it...**

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

**reprint it...**

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

**RePrints**

### WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altworks	www.altworks.com	603-598-2582	52
BEA Systems	http://dev2dev.bea.com/useworkshop	800-817-4BEA	41
BEA dev2dev Days	www.bea.com/events/dev2devdays	404-240-5506	11
Borland	www.borland.com	831-431-1000	2
Covasoft	www.covasoft.com/today's_tech/	888-963-2682	13
Dirig Software	www.dirig.com/illusion/weblogic	603-889-2777	9
Intel	www.intel.com/ad/bea	408-765-8080	29
PANACYA	www.panacya.com	877-726-2292 x3344	3
Performant	www.performant.com/weblogic1	866-773-6268	37
Precise Software Solutions	www.precise.com/wldj	800-310-4777	15
Rational Software	www.rational.com/offer/bea	800-728-1212	19
Sitraka	www.sitraka.com/jclass/wldj	800-663-4723	17
Sitraka	www.sitraka.com/performance/dev2dev	800-663-4723	25
Sitraka	www.sitraka.com/performance/wldj	800-663-4723	51
SYS-CON Publications	www.sys-con.com	201-802-3020	31
SYS-CON Reprints	www.sys-con.com	201-802-3026	47
Web Services Edge World Tour	www.sys-con.com	201-802-3069	45
Web Services Journal	www.wsj2.com	888-303-5282	49
Web Services Resource CD	www.jdjstore.com	201-802-3012	27
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	33
Wily Technology	www.wilytech.com	888-GET-WILY	4

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.

# LOOK WHAT'S COMING NEXT MONTH

### PeopleSoft Integration Using Web Services

This month Joe Krozak talked about integrating PeopleSoft with JMX. In November, he'll look at integrating PeopleSoft with one of the newest technology paradigms: Web services

### Administrative Tasks

Dr. Subrahmanyam Allamaraju provides several ways to configure servers, clusters, machines, JDBC connection pools, and JMS servers using the weblogic.Admin Command-Line Utility

### Programmatic Clients, Symmetry and The Humble Ant

In his Transactions Management column, Peter Holditch looks at one of the pieces of arcane about the WebLogic Server transaction manager implementation

### Plus, BEA WebLogic Developer's Journal presents several case studies from companies using this platform:

- \* A financial services company that needed more sophisticated features and functions decided to move to their applications to BEA WebLogic
- \* A medical company needed to deploy the client-side portion of their BEA WebLogic application to 10,000 users and had to choose a new deployment solution
- \* An automobile collision repair firm streamlines and automates data exchanges – shifting from outdated manual processes based in proprietary, legacy systems to collaboration over Internet channels
- \* The application server leader achieves business-focused operational excellence and maintains not only its technology leadership but also its continued leadership in serving customers and business partners





under-perform compared with the in-memory replication (see Figures 4, 5, and 6). JDBC-based session persistence caters more toward those applications that require long-term persistent storage of session data and is ideal for situations where failover and data-persistence requirements take preference over performance.

### In-Memory-Based Session Persistence

In contrast to the previously described persistence mechanisms, in-memory-based session persistence requires the use of WebLogic clustering and offers the best overall choice in terms of performance and other tradeoffs that may be required in a typical

real-world situation. To configure in-memory session persistence, set the PersistentStoreType property in the <session-descriptor> element of weblogic.xml to *replicated*. In-memory replication is achieved by maintaining a replicated session or in-memory replica of the session state on the the secondary server in a clustered configuration. By default, WebLogic Server attempts to create session state replicas on a different machine than the one that hosts the primary session state. You can further control (via the WebLogic Server console, for instance) where the secondary states are placed using replication groups. A replication group is a preferred list of clustered servers to be used for storing session state replicas.

The primary server handles the user's requests while the WebLogic replication manager creates a secondary server for this user session and maintains a copy of the session object. The replication manager handles all the details of replicating the session data to the remote secondary and keeps this copy in sync with the primary server's session object.

The following Web server and proxy software support in-memory replication for HTTP sessions:

- WebLogic Server with the HttpClusterServlet
- Netscape Enterprise Server with the Netscape (proxy) plug-in
- Apache with the Apache Server (proxy) plug-in
- Microsoft Internet Information Server with the Microsoft-IIS (proxy) plug-in

Alternatively, load-balancing hardware may be used instead of these proxy plug-ins. The proxy plug-in (or the load balancer) used between the Web application and the client is configured to fetch the information about the primary and secondary servers in the cluster from the client request. In the event of a primary server failure, the client's request is moved to the secondary server. This now acts as primary server and a new secondary is created for backup. In the event of a secondary server failure, a new secondary is created and the primary server is updated for this change. All of these details are completely transparent to the user. The in-memory-based session persistence mechanism provides excellent performance (see Figure 5) and reliable failover capabilities.

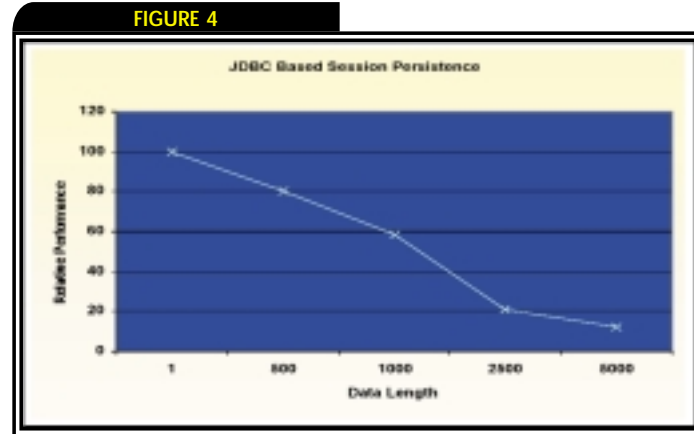
### Summary

The various session persistence mechanisms have been designed with a view to meeting a wide-ranging set of problem domains and user requirements. A proper understanding of these mechanisms (and the respective tradeoffs) is essential in deciding on a solution that is best suited for your application. While the graphs shown in Figures 1-5 indicate the impact of data size on the performance of the various session persistence mechanisms, Figure 6 shows the performance for the different session persistence mechanisms using a session data length of 1000 for a StringBuffer object. Since the file-based mechanism has the worst performance, all results are shown relative to this result.

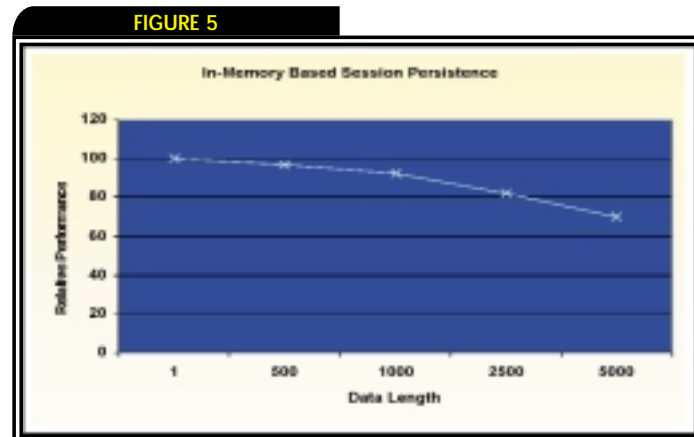
### References

Further information on configuring WebLogic Server 7.0 to use the different session persistence modes is available at [http://e-ddocs.bea.com/wls/docs70/web\\_application/sessions.html#session-persistence](http://e-ddocs.bea.com/wls/docs70/web_application/sessions.html#session-persistence).

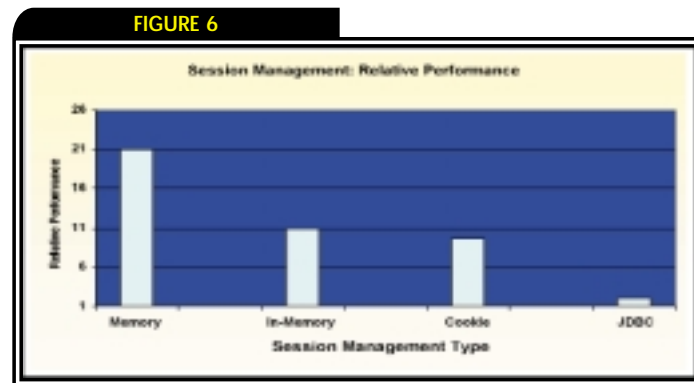
In addition, the white paper "Achieving Scalability and High Availability for E-Business" ([www.bea.com/products/weblogic/server/paper\\_wls\\_clustering.pdf](http://www.bea.com/products/weblogic/server/paper_wls_clustering.pdf)) presents an overview on the various clustering capabilities of BEA WebLogic Server.



JDBC-based session persistence



In-memory-based session persistence



Session Management: Relative Performance

# WebServices JOURNAL

.NET J2EE XML

LEARN WEB SERVICES. GET A NEW JOB !

## Subscribe today to the world's leading Web Services resource

The Best .NET Coverage Guaranteed!

Get Up to Speed with the Fourth Wave in Software Development

- Real-World Web Services: XML's Killer App!
- How to Use SOAP in the Enterprise
- Demystifying ebXML for success
- Authentication, Authorization, and Auditing
- BPM - Business Process Management
- Latest Information on Evolving Standards
- Vital technology insights from the nation's leading Technologists
- Industry Case Studies and Success Stories
- Making the Most of .NET
- Web Services Security
- How to Develop and Market Your Web Services
- EAI and Application Integration Tips
- The Marketplace: Tools, Engines, and Servers
- Integrating XML in a Web Services Environment
- Wireless: Enable Your WAP Projects and Build Wireless Applications with Web Services!
- Real-World UDDI
- Swing-Compliant Web Services
- and much, much more!

Only \$69.99 for 1 year (12 issues)\*  
 \* Newsstand price \$83.88 for 1 year  
 Subscribe online at [www.wsj2.com](http://www.wsj2.com) or call 888 303-5252  
 \*Offer subject to change without notice





# News & Developments

## Palm and BEA Partner to Mobilize Web Services in the Enterprise

(Milpitas, CA and San Jose, CA) – Palm Solutions Group, a pioneer in mobile and wireless Internet solutions and a provider of handheld computers; and BEA Systems, Inc., the world's leading application infrastructure software company, have announced a long-term, strategic relationship to bring Web services to Palm-branded handhelds.



Palm will develop tools and a device-side software suite to simplify development and deployment of mobile Web services to extend enterprise applications on Palm handhelds. It will work with BEA to integrate this solution with BEA WebLogic Server 7.0 and BEA WebLogic Workshop products as server-side controls. This will be the first WebLogic Workshop control developed for a handheld device. [www.bea.com](http://www.bea.com), [www.palm.com](http://www.palm.com)

## PeopleSoft Launches Application Integration Product Suite

(New Orleans) – PeopleSoft, Inc., has announced PeopleSoft AppConnect, a new product suite of preintegrated portal, integration, and warehouse solutions that reduces the complexities of integrating multivendor applications. PeopleSoft AppConnect utilizes Web services technology to simplify integration.



The product suite includes the Enterprise Portal, Integration Broker, and Enterprise Warehouse. Products communicate using Web services and legacy integration methodologies. Additionally, AppConnect

is compatible with J2EE, and works with BEA WebLogic application servers. [www.peoplesoft.com](http://www.peoplesoft.com)

## Evolution Hosting Offers WebLogic in Flagship Platform

(Boulder, CO) – Evolution Hosting, a provider of J2EE data center management, now supports WebLogic Server in its core offering. The Evolution Hosting Platform automatically configures application, database, and Web servers in a data center, alleviating common production issues that plague enterprise Java environments such as account



setup, resource connectivity, security, and management of server properties. Evolution Hosting also offers managed application hosting for companies wishing to outsource their J2EE application management. [www.evolutionhosting.com](http://www.evolutionhosting.com)

## Alignment Software Releases AppAssure

(Superior, CO) – Alignment Software (formerly known as NPULSE Software), makers of next-generation performance management software for complex, distributed-application infrastructures, has announced the general availability of its AppAssure Application Performance Management Software. AppAssure delivers rapidly deployable, dynamic, tunable application monitoring to operations staff who are stretched to the limit by increasingly complex distributed applications.



Users have a real-time view from URL to SQL, and from the system level down to individual methods within an application. Users can selectively choose to monitor URL transactions, or

manage granular data down to the method call within their applications. The depth of monitoring can be changed dynamically in response to system performance thresholds. [www.alignmentsoftware.com](http://www.alignmentsoftware.com)

## German Air Force Signs Agreement with BEA Systems

(San Jose, CA) – BEA Systems, Inc., has signed an agreement with the German Air Force to develop standards-based application server technology from BEA for a military command and control system that could become a model for all NATO countries.



The Baltic Command and Control Information System (BALTCCIS) is being developed by the German Air Force as part of its bilateral cooperation with countries that might join NATO in the future. The system will provide a shared data pool and deliver the necessary information for critical command decisions about the deployment and allocation of shared resources.

Once complete, the system will be divided into two functional areas: Command and Control and Common Support. Command and Control will use BEA WebLogic Server as the platform for industry-standard Enterprise JavaBeans-based business logic, and a Microsoft SQL database server. [www.baltccis.net](http://www.baltccis.net)

## Objectivity/DB Release 7.1 Delivers Enhanced Interoperability

(Mountain View, CA) – Objectivity, Inc., a provider of scalable high-performance databases for the management of complex data, has announced the general availability of Objectivity/DB Release 7.1.

Objectivity/DB Release 7.1 provides transparent interoperability for C++, .NET, Java, Smalltalk, and SQL++/ODBC applications running on AIX, HP-UX, IRIX, LynxOS, Red Hat Linux, Solaris, Tru64 UNIX, and Windows. The BEA-ready release adds container-managed persistence to its existing WebLogic-ready facilities. [www.objectivity.com](http://www.objectivity.com)



## Ceon Elaborates on Successful Partnership with BEA

(Redwood City, CA) – Ceon Corp. has announced that it fully utilizes the BEA WebLogic Enterprise Platform in its service fulfillment solution. By incorporating BEA Tuxedo, BEA WebLogic Server, and BEA WebLogic Integration into Ceon Integration and Provisioning Suite (Ceon IPS 4.0), it delivers unprecedented speed to revenue and ROI to the communications services market.

Ceon IPS 4.0 activates new services, manages order workflow, and enables subscriber Web self-service. It offers customers competitive advantages by lowering operating expenses, generating revenue faster, and creating the flexibility to bundle multiple services for higher profitability. [www.ceon.com](http://www.ceon.com)



# Sitraka

[www.sitraka.com/performance/wldj](http://www.sitraka.com/performance/wldj)

# Altaworks

[www.altaworks.com](http://www.altaworks.com)