

LET YOUR APPLICATION BE HEARD

BEA WebLogic

DEVELOPER'S JOURNAL

NOVEMBER 2003 - Volume:2 Issue:11

weblogicdevelopersjournal.com

International Conference & Expo



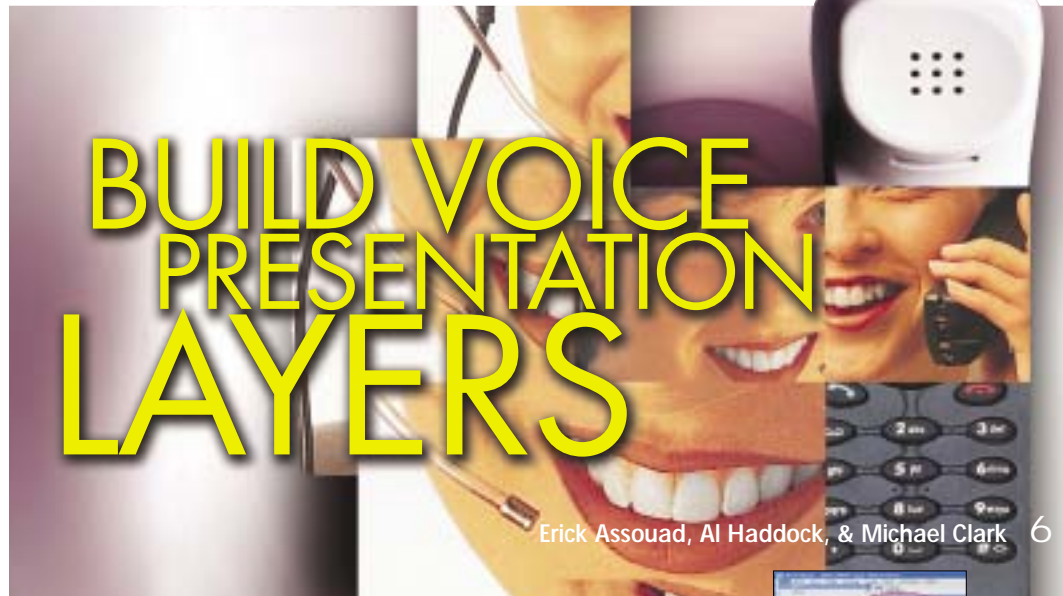
Development Technologies Exchange
February 24-26, 2004
 Hynes Convention Center, Boston, MA

ARCHITECTING JAVA, .NET, WEB SERVICES, OPEN SOURCE, AND XML

- Application Integration
- Backup Java
- Mobility
- SOAP
- XML
- JavaServer Faces
- SOA
- Interoperability
- Java Training

Register by November 21, 2003
SAVE \$400

see page 31 for details



BUILD VOICE PRESENTATION LAYERS

Erick Assouad, Al Haddock, & Michael Clark 6

FROM THE EDITOR

Tower of Babel
 by Joe Mitchko
 page 4

TRANSACTION MANAGEMENT

Transactions: How Big Are Your Atoms?
 by Peter Holditch
 page 32

NEWS & DEVELOPMENTS
 page 50

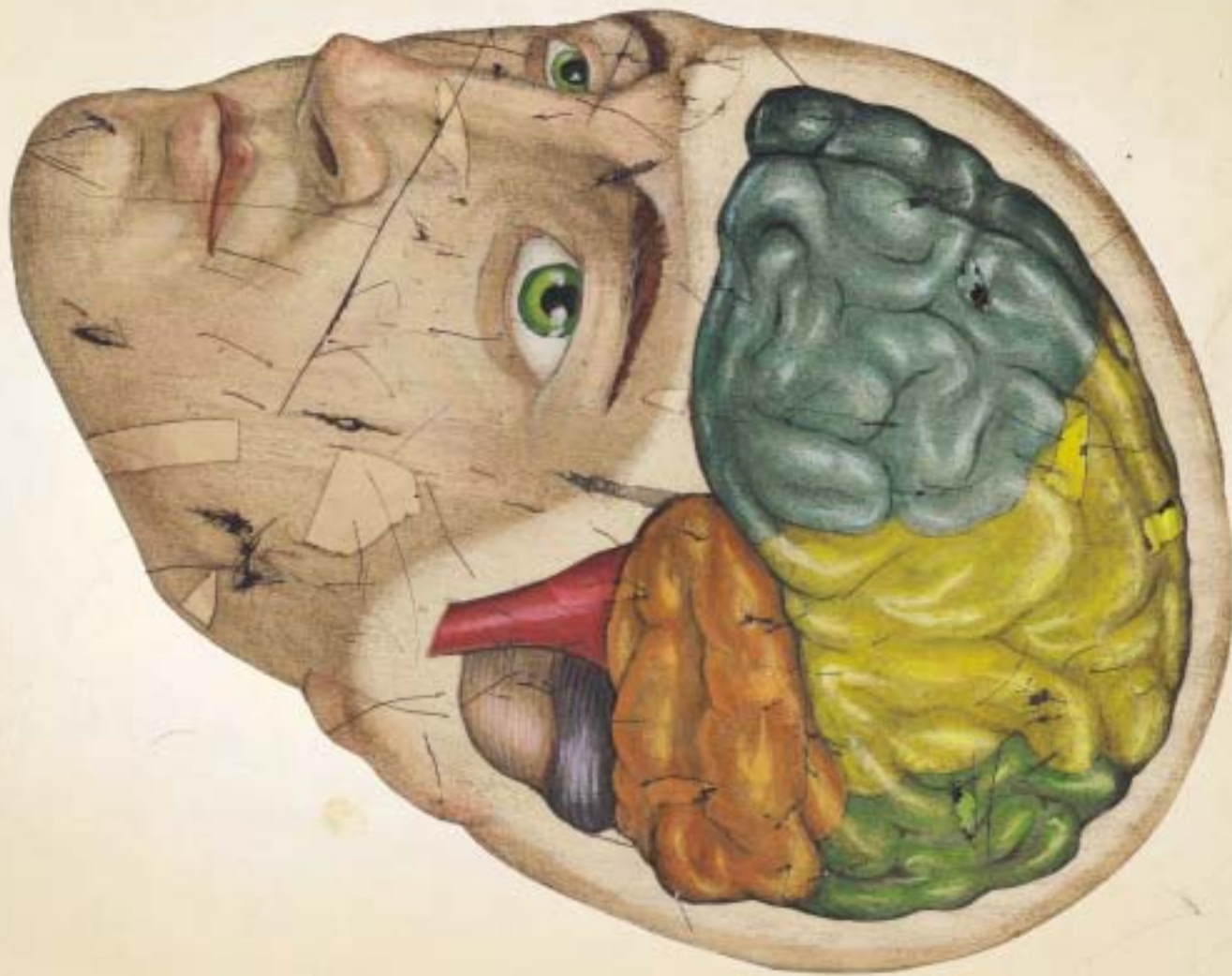
RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2004

\$8.99US \$9.99CAN



SYS-CON MEDIA

ADMINISTRATION: Application Management with WebLogic Server for Developers PART 2 Application deployment and monitoring tools		14	Vadim Rosenberg & Robert Patrick
SERVER: Automated WebLogic Server Domain Setup Saving time in a complex environment		20	Andreas Wittmann
SERVER: Parallel Business Logic Processing A reusable framework can solve problems		26	Murali Kashaboina & Bin Liu
SECURITY: How to Secure a Web Application WebLogic Server extensions can dovetail with J2EE		34	Neil Smithline
MANAGEMENT: Management and Monitoring Using the JRockit JVM One key to better problem diagnosis		40	Kunal Mittal
PRODUCT REVIEW: Cyanea/One from Cyanea An out-of-the-box total production monitoring solution		42	Jason Snyder
PERFORMANCE: How to Diagnose a Performance Problem in a J2EE System The cure may be simpler than you think		44	John Bley



DO YOU HAVE A BRAIN?

CEREBRUM: CONTROLS THOUGHT.

*What is 2+2? What color is the sky?
If you can answer these, then you have a cerebrum.*

MEDULLA OBLONGATA: CONTROLS INVOLUNTARY FUNCTIONS.

*Check your pulse. Do you have one?
Then you have a medulla oblongata, too.*

BROCA'S AREA: CONTROLS LANGUAGE.

*Say the following sentence: "Frankly, Hector, I'm a bit surprised."
Did it work? Then you have a Broca's Area.*

PITUITARY GLAND: CONTROLS HORMONES.

*When you were about 13, did your voice change?
Did you grow hair in special places?
Then you've got a pituitary gland, friend.*

YOU HAVE A BRAIN.

So why spend so much time on mindless coding?

Download Weblogic Workshop Now.

BEA WebLogic Workshop™ 8.1 is a more efficient way to develop in J2EE.
And that means less grunt work and more real work. dev2dev.bea.com/medulla.





EDITORIAL



BY JOE MITCHKO

Tower of Babel

I currently find myself on a consulting engagement for a large, multimillion-dollar, enterprise-wide Web services project for a major Fortune 500 firm. It's a golden opportunity to see first-hand the development of a bleeding-edge enterprise service bus (ESB), complete with hundreds of Web services-enabled legacy systems and a sophisticated call center workstation front end.

One of my responsibilities is to meet with all of the various application and system development teams across the enterprise and assist them in achieving the final goal, which is deploying all of the various business services, based on ported applications, to production. Here is where the fun begins. Such a simple statement, "deploy" it to production, except that in this new service-oriented architectural world, "it" means different things to different people. So does the term "production" for that matter.

In the good old days of systems development, deploying applications was straightforward. You had requirements, a design; you coded and tested, and when it worked, you moved it into production. Even with online e-commerce Web applications, this formula held true. Not any more, especially at the enterprise level.

It's as if everyone started speaking their own language, and long-held project terminology, the glue that holds together large development initiatives, no longer seems able to do the job. The application? Which one? Can a legacy application still be considered an application if its business logic is exposed through a series of Web service interfaces?

The application is currently scheduled to go through testing. Wait a minute, how can that be when it has been in production for years? Oh, I see, we're talking about the Web service interface tied to the business-logic tier. The current plan is to roll out a pilot version to production in the fourth quarter this year, so we need to get

the operations staff ready. Whoa, hate to point this out but we already have the operations staff supporting a handful of Web services plugged into the ESB in production with active users. Yes, but we're talking about the final deployment initiative centered on getting the call center workstation online and into production along with the remaining migrated applications. You know, the real push to get the "system" into production. System? Almost overnight, it seems that basic project terminology has become somewhat outdated.

To add to the confusion, the integration issues that come up on such an initiative require unprecedented levels of cooperation among development teams across the enterprise, teams that were never required to work together in the past, due to the stovepipe development mentality of each business owner. Moreover, it's simplistic to say that all you have to do is Web service-enable your legacy applications (made increasingly easy by development tools such as WebLogic Workshop) and then they can be used in portals or process flows. In reality, all applications being ported over will need to conform to a standardized framework for the handling of instrumentation, exception handling, user provisioning, security, and data (to name a few).

Dealing with this level of change across the enterprise makes the technology part look simple. Over time, people will catch up with the terminology, but the coordination and integration of the various Web service-enabled applications across the enterprise in a service-oriented architecture requires more sophisticated configuration management tools along with support for enterprise-wide development, testing, and deployment. The more assistance BEA and other vendors can provide to help manage the complexity by including enterprise-wide workgroup support and configuration tools in their development tools, the better. ●

AUTHOR BIO...

Joe Mitchko is the editor-in-chief of WebLogic Developer's Journal and a senior technical specialist for a leading consulting services company.

CONTACT: joe@sys-con.com

BEA WebLogic

DEVELOPER'S JOURNAL

EDITORIAL ADVISORY BOARD
TOM ANGELUCCI, LEWIS CIRNE, STUART HALLOWAY,
KEVIN JONES, TYLER JEWELL, WAYNE LESLEY LUND,
SEAN RHODY, CARL SJOGREEN, BLAKE STONE

FOUNDING EDITOR

PETER ZADROZNY

EDITOR-IN-CHIEF

JOE MITCHKO

PRODUCT REVIEW EDITOR

JASON SNYDER

EXECUTIVE EDITOR

GAIL SCHULTZ

EDITOR

NANCY VALENTINE

ASSOCIATE EDITORS

JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR

JENNIFER VAN WINCKEL

WRITERS IN THIS ISSUE

ERIC ASSOUAD, JOHN BLEY, MICHAEL CLARK,
AL HADDOCK, PETER HOLDITCH,
MURALI KASHABOINA, BIN LIU, JOE MITCHKO,
KUNAL MITTAL, ROBERT PATRICK,
VADIM ROSENBERG, NEIL SMITHLINE,
ANDREAS WITTMANN

SUBSCRIPTIONS

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department,
SUBSCRIPTION HOTLINE:

888-303-5282

Cover Price: \$8.99/Issue

Domestic: \$149/YR (12 Issues)

Canada/Mexico: \$169/YR

Overseas: \$179/YR

(U.S. Banks or Money Orders)

PRESIDENT AND CEO

FUAT KIRCAALI

VP, BUSINESS DEVELOPMENT

GRISHA DAVIDA

GROUP PUBLISHER

JEREMY GEELAN

TECHNICAL DIRECTOR

ALAN WILLIAMSON

PRODUCTION CONTROLLER

JIM MORGAN

ART DIRECTOR

ALEX BOTERO

ASSOCIATE ART DIRECTORS

LOUIS F. CUFFARI • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR

TAMI BEATTY

SENIOR VP, SALES & MARKETING

CARMEN GONZALEZ

VP, SALES & MARKETING

MILES SILVERMAN

ADVERTISING SALES DIRECTOR

ROBYN FORMA

DIRECTOR OF SALES AND MARKETING

MEGAN MUSSA

ADVERTISING SALES MANAGER

ALISA CATALANO

ASSOCIATE SALES MANAGERS

CARRIE GEBERT • KRISTIN KUHNLE

PRESIDENT, SYS-CON EVENTS

GRISHA DAVIDA

CONFERENCE MANAGER

MICHAEL LYNCH

NATIONAL SALES MANAGER

SEAN RAMAN

FINANCIAL ANALYST

JOAN LAROSE

ACCOUNTS RECEIVABLE

KERRI VON ACHEN

ACCOUNTS PAYABLE

BETTY WHITE

VP, INFORMATION SYSTEMS

ROBERT DIAMOND

WEB DESIGNERS

STEPHEN KILMURRAY • CHRISTOPHER CROCE

ONLINE EDITOR

LIN GOETZ

CIRCULATION SERVICE COORDINATORS

NIKI PANAGOPOULOS • SHELLIA DICKERSON

EDNA EARLE RUSSELL

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

WORLDWIDE NEWSSTAND DISTRIBUTION

Curtis Circulation Company, New Milford, NJ

FOR LIST RENTAL INFORMATION:

Kevin Collopy: 845 731-2684, kevin.collopy@edihroman.com

Frank Cipolla: 845 731-3832, frank.cipolla@eposidirect.com

© COPYRIGHT 2003 BY SYS-CON PUBLICATIONS, INC. ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM, WITHOUT WRITTEN PERMISSION. FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR, SYS-CON PUBLICATIONS, INC. RESERVES THE RIGHT TO REVISE, PUBLISH AND AUTHORIZE THE READER TO USE THE ARTICLES SUBMITTED FOR PUBLICATION. ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES. SYS-CON PUBLICATIONS, INC. IS NOT AFFILIATED WITH THE COMPANIES OR PRODUCTS COVERED IN WEBLOGIC DEVELOPER'S JOURNAL.



© 2003 Wily Technology, Inc. The Wily logo is a trademark of Wily Technology, Inc. Java is a trademark of Sun Microsystems in the U.S. and other countries.

Two years without a vacation.
The application's up. It's down.
It's up. It's down.

I'm to blame. Steve's to blame.
Someone's always to blame.



Not any more.

Get Wily.™

*Enterprise Java
Application Management*
1 888 GET WILY
www.wilytech.com

wily
technology



BY ERIC ASSOUD,
AL HADDOCK & MICHAEL CLARK

AUTHOR BIO...

Eric Assouad is a senior software engineer for SandCherry with nearly a decade of experience developing both ERP and telecommunications object oriented software. He has held key architect, developer, and software team lead roles.

Al Haddock is a senior software engineer with SandCherry and has over 15 years' experience developing software for the defense, bioinformatics, and telecom industries.

Michael Clark is a software engineer for SandCherry experienced in developing speech and multi-modal (data+voice) applications using VoiceXML and Speech Application Language Tags (SALT).

CONTACT...

eassouad@sandcherry.com
ahaddock@sandcherry.com
mclark@sandcherry.com

SandCherry, Inc. (www.sandcherry.com) is a provider of software for developing, testing, and deploying advanced speech-enabled services.

BUILD VOICE PRESENTATION LAYERS FOR DELIVERING APPLICATIONS TO PHONES

EXTENSIBILITY THAT MEETS A UNIQUE

BEA WebLogic Workshop 8.1 provides a wide range of tools for creating Web server applications. Components integrated using Workshop's extensible component model include Java controls, page flows, and source and design views that are available to any software vendor or Web application developer.

As the foundation of an integrated development environment, WebLogic Workshop offers the power to build new Web services that naturally inherit its Web design innovations. SandCherry's AppDev VXML is one example of extending BEA WebLogic Workshop 8.1 functionality to allow the creation of services that can be accessed by phone.

Interactive Voice Response (IVR) and speech-based services using touch-tone or speech recognition can now be consolidated with Web applications into a three-tiered application infrastructure, reducing solution costs and improving the user experience. These services have traditionally run on proprietary silos of hardware and software completely independent of Web-based applications and infrastructure. The silo approach drives up solution costs by duplicating data integration, business logic, and systems – not to mention the cost of user dissatisfaction caused by inconsistent

experiences between different voice and Web systems. Developers can now simply add a new voice presentation layer to Web applications to extend their reach to any customer or employee with a phone while reusing existing business logic and back office resources (see Figure 1).

The SandCherry AppDev voice extension facilitates development of the voice presentation layer for controlling speech and media resources, which include automatic speech recognition (ASR), text-to-speech (TTS), and prerecorded prompts. By integrating the voice presentation layer development components into WebLogic Workshop 8.1, developers can reduce the time and cost of developing and maintaining a speech application by reusing the controls and models already created in the Workshop IDE for a Web application.

Integration Overview

Any voice application requires three components to work: the voice user interface design and flow, written in VoiceXML (VXML); grammars that identify what spoken input the application will recognize, written in Grammar XML (GRXML); and audio prompts and announcements played to the user that are captured as wave or other types of audio formats. Adding a voice application development environment under WebLogic





CHALLENGE

Workshop required integration with five major Workshop components to provide a seamless development experience between Web and voice Web projects. Extending these components consists of integrating previously stand-alone applications and creating new applications that dock and run within the Workshop workspace. The five components extended by AppDev VXML and the tools that use them are:

- **Web Project:** Voice Web Project
- **Document:** Audio, Grammar XML (GRXML), and VoiceXML (VXML) document types
- **Tag Library:** VoiceXML and GRXML tags for voice documents
- **Action:** Voice-related menu and toolbar and launching external applications
- **Frame:** Application frames that can be docked within WebLogic Workshop

The functionality added to WebLogic Workshop offers both development and testing tools, since one of the primary objectives in building the development environment was to create a complete stand alone development and test environment for speech applications. Using the

Workshop-based approach, developers no longer require an expensive and complex speech hardware platform to develop and test voice applications.

Voice Application Development Tools

Voice applications require several different components that must be combined to deliver the three basic functions required for a voice interaction with a user: controlling the application flow, processing voice input from a user, and outputting prerecorded (or synthesized) information to a user. The development tools added represent the combination of the Web view presented by Workshop and the voice application view required by the speech and media components used to deliver the voice application. The development tools added include:

- **Voice Web Project:** Project to maintain voice-related documents and functionality. A Voice Web Project inherits the Web project functionality of a WebLogic Workshop Web Project.
- **VXML Document:** Document containing a VXML page depicting the flow of a voice application. A VXML document inherits JSP document functionality from the WebLogic Workshop JSP Document.
- **GRXML document:** Document containing a GRXML page, which lists the valid text (grammars) representing the spoken words the application will recognize. A GRXML document inherits JSP document functionality in the same manner as a VXML document.
- **Audio Document:** Document containing wave audio data of a prompt or announcement to play to a caller. The prompt is generally played before trying to recognize a caller's response. This is a completely new document type that extends Workshop's base Document class.
- **Tree Design View:** Creates a tree-oriented view of a VXML or GRXML page. The tree gives the developer an easy-to-manage and readable view of the voice documents just as Workshop does for Web documents.
- **Voice Menu and Toolbar:** Menu and toolbar containing voice application tools. SandCherry added the Log Viewer, Validator, SoftPhone, and Test Voice Application tools that are available from the voice menu and toolbar. These options were incorporated by extending the WebLogic Workshop Action class.
- **Validator:** Uses a third-party parser to validate a VXML or GRXML page being hosted by the WebLogic Web Server. Integrating this component required extending the WebLogic Workshop Frame class.

Together, these tools allow a developer to add tags from VXML and GRXML tag libraries, create wave files to be played within a voice application, and validate the scripts of the application before testing.

Voice Web Project

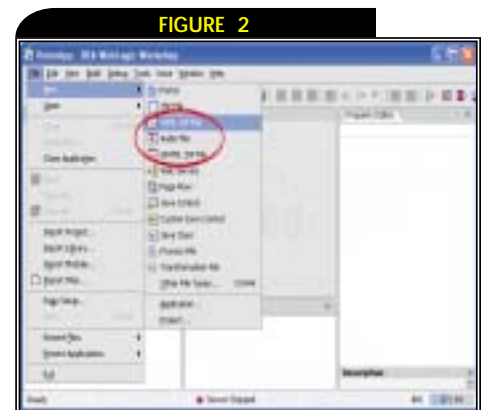
The Voice Web Project is a new project type created to hold both voice and Web-related documents. The project extends WebLogic Workshop's Web Project, and by doing so inherits Web project functionality. Java Controls (JCX), Java Page Flows (JPF), and any other legal Web document may be added to a voice project. Once a new application has been created, the "Voice Web Project" can be found under the "Voice Interface" category. This project type was created to accommodate the integration of VXML, GRXML, and audio documents for a Voice User Interface (VUI) into the Workshop model while providing sufficient separation due to the unique nature of speech applications to avoid unnecessarily complicating existing Web projects in Workshop.

The Documents

There are three new voice document types that exist only within a Voice Project - VXML, GRXML, and audio documents. Once the project has been created, a developer can create these voice documents just like any other Workshop document (see Figure 2).

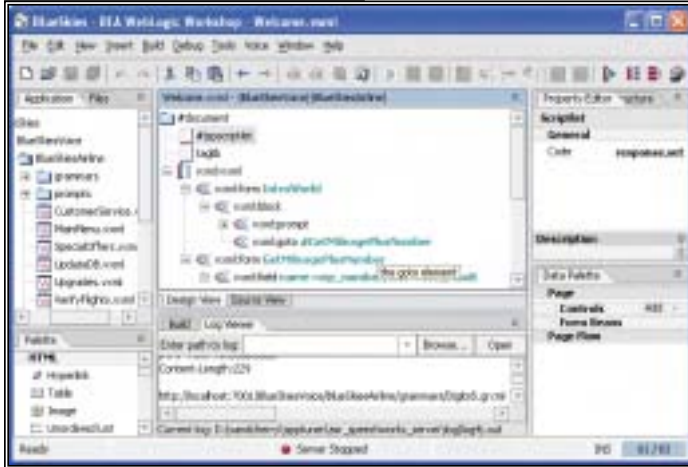


Adding a voice presentation layer to Web applications



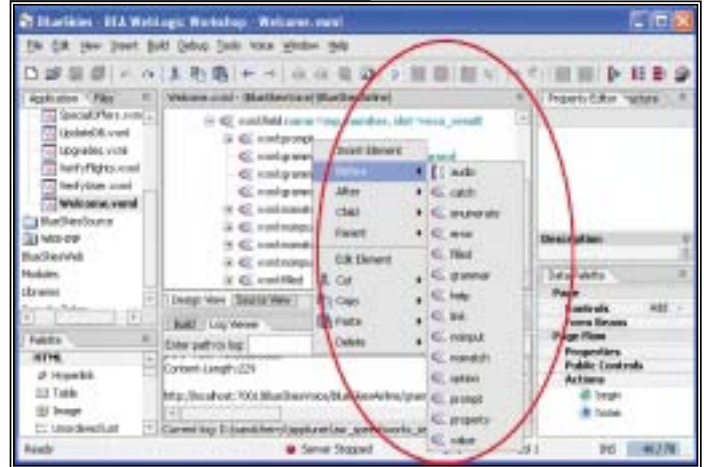
Voice document created like a WebLogic Workshop document

FIGURE 3



Source editor

FIGURE 4



Tree Design view

VXML DOCUMENT

Voice Extensible Markup Language (VoiceXML or VXML), defined by the W3C standards body, is the leading speech application language today. The description of VXML contained in the standard is:

“VoiceXML is designed for creating audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and DTMF key input, recording of spoken input, telephony, and mixed initiative conversations. Its major goal is to bring the advantages of Web-based development and content delivery to interactive voice response [IVR] applications.”

The SandCherry team identified several overriding goals that guided the WebLogic Workshop extensions to support VXML files:

1. WebLogic Workshop must understand VXML 2.0 in order to provide familiar IDE development aids such as property editing and code completion.
2. HTML and VXML document types must be different within Workshop.
3. Each VXML file must be processed as a JavaServer Page (JSP) to harness the power of the BEA WebLogic platform.

Since SandCherry’s AppDev’s VXML was developed in parallel with the WebLogic Workshop integration, fundamental design choices such as using JSPs for the VXML pages within AppDev allowed the development team to easily integrate with Workshop’s JSP Document Class and inherit the capabilities provided by WebLogic Workshop. Creating a custom JSP tag library for VXML made it possible to integrate into Workshop’s JSP designer and

reuse many of its features to help developers write VXML. Because some functionality must be enabled when an AppDev document is open, but disabled when a standard Workshop document is open, a new document type was required. Creating the new document type was accomplished within the WebLogic Workshop extensibility framework by extending the base Document class to create a new “voice document” type for both VXML and GRXML file types.

GRXML DOCUMENT

GRXML is another W3C standard used for defining grammars within voice applications. As stated in the specification:

“Grammars [are] for use in speech recognition so that developers can specify the words and patterns of words to be listened for by a speech recognizer. The syntax of the grammar format is presented in an XML Form.”

GRXML documents were added to WebLogic Workshop in the same way as VXML documents, with the “voice document” type mapped to the file extension “grxml”. A unique tag library was created for GRXML based on W3C’s “Speech Recognition Grammar Specification” (SRGS 1.0).

AUDIO DOCUMENT

Audio documents present two unique challenges within the WebLogic Workshop framework since an audio document is simply a wave file used within a voice application. First, audio files include wave recordings, not text, yet must still reside in the Workshop project so that the applica-

tion can use them. The second challenge is that presenting an application developer with a GUI-based source or design view of a wave file isn’t very useful. An audio recording tool must be embedded into the Workshop document editor to address these challenges.

BEA WebLogic Workshop offers a clean and easy solution to these challenges by giving the tool developer the flexibility to extend the base Document class, remove the source view, and embed a once stand-alone tool into the Workshop workspace. Using this method, an audio recording and playback tool was integrated into the Workshop environment so that when an audio document is opened, the audio tool is launched in what is normally the source view window. This tool records audio files for use during development and testing of the voice application.

Tree Design View

A Tree Design view was created for VXML and GRXML documents to take the place of the original Design View. Since both VXML and GRXML documents inherit JSP functionality, the original Design view was not useful for creating voice applications. Using a tree to display and develop a voice application provides a clear hierarchical view of the script. Under this view, a developer may also close and open nodes to shrink and expand sections of the application to focus on particular portions of the document. WebLogic Workshop provides the flexibility to disable the original view and create a new one that can be displayed as a tab in the source editor (see Figure 3).

With all of the functionality it provides, the Tree Design view is a very powerful tool for creating a voice application. First, it

Under pressure to improve e-business performance

Struggling to support internal customers

Worried about delivering and measuring ROI



Is the burden of e-business application performance on your shoulders?

E-business is more demanding than ever on IT departments. You have to stabilize, manage and optimize the e-business infrastructure to ensure maximum availability, rapid recovery, and minimal transaction loss. You must support the unique needs of internal customers. Drive down costs. Deliver and measure ROI. And, ultimately, get the most out of your substantial investments in J2EE.

Let this FREE audio CD lighten the load.

Request your complimentary copy of **Advancing e-Business Performance: Five Ways to Achieve Greater Manageability and ROI**. It's a candid, insightful discussion between leading e-business technology experts and industry analysts, along with senior corporate IT executives who are successfully addressing the same e-business challenges you face.

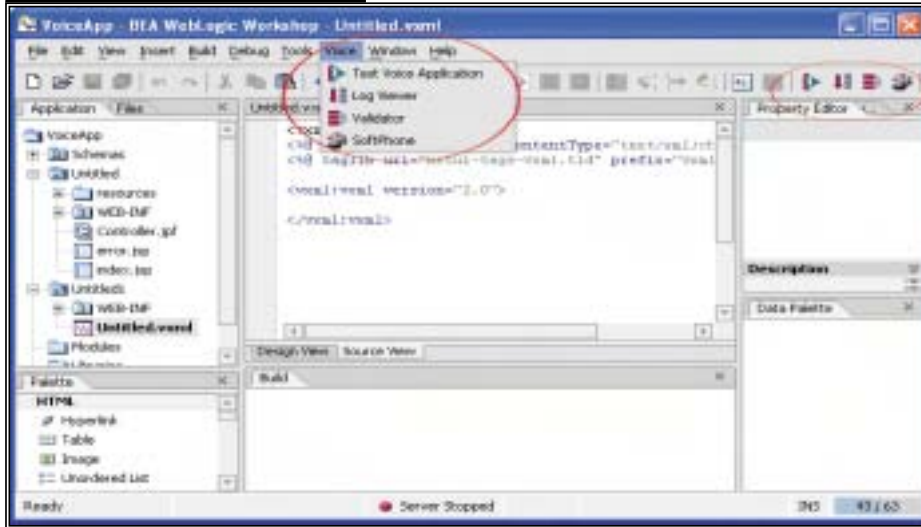


It's about stabilizing, managing and optimizing e-business application performance. Gaining actionable business insight from performance data. Serving the unique needs of your internal business customers. Calculating, delivering and measuring ROI. And getting the most out of your J2EE investments.

To request your FREE copy, visit
www.ebizperform.com



FIGURE 5



Voice application

FIGURE 6



SandCherry SoftPhone

provides integral error checking of the tags. Second, it makes it easy to cut, paste, copy, and move large sections of a document. Finally, it provides an easy-to-read view of the entire voice document.

The error checking built into the Tree Design view offers users a familiar environment to build voice files incorporating many of the features found in the normal WebLogic Workshop design view: nodes containing errors are color coded in a similar fashion; right-clicking on elements (tree nodes) within the view displays a menu that includes the legal actions for that node; and users can drag-and-drop any tag for any node. Additional intelligence is provided on insertion to ensure that tags are valid within the VXML specification to help avoid basic errors. The Tree Design view uses the underlying Workshop compiler service to provide diagnosis for each of the nodes (see Figure 4).

Voice Menu and Toolbar

The Voice Menu and Toolbar are simple controls that allow a user to start voice tools, including Test Voice Application, Log Viewer, Validator, and SoftPhone.

Adding these tools to the menu options is easily accomplished by extending the

Workshop Action class with several lines of XML that specify the location of the tool, the icon, and the Java code to run when the option is selected. A snippet of script for adding the Validator menu option is shown in Listing 1.

In this example, the “action class” is the Java code to launch the Validator, Validator.gif is the icon to display, and “location priority” specifies where to display the tool within the toolbar and menu. Adding actions within WebLogic Workshop is simple, yet powerful, since these actions integrate seamlessly and inherit the Workshop look and feel (see Figure 5).

Validator

The Validator tool verifies VXML and GRXML documents by pulling both types of files from a WebLogic Server and running them through a “validating parser” (Apache Xerces) in AppDev VXML using the W3C Schema appropriate for the current document. The parser downloads the active file in the same way that a VXML browser downloads applications from the Web server during service deployment. The content being validated is displayed in a window, along with a list of any errors found in the document.

The Validator, which is written in Java, uses a Swing panel that plugs neatly into Workshop’s Frame extensions. These extensions allow the Validator to provide a consistent UI experience for the developer by inheriting Workshop’s look, feel, and functionality.

Test Voice Tools

Testing a voice application offers a new set of challenges not originally envisioned within the Workshop environment. Where a Web application can be easily displayed in a window, a voice application requires a phone-based interaction with a speech platform controlling a number of different media components. SandCherry has pre-integrated a variety of components using its software-based speech platform and added a software-based phone emulator to provide a complete test environment for voice applications. Four tools are used when testing a voice project within an application.

- **Test Voice Application:** This component starts the voice platform and component processes necessary for testing a voice application.
- **SoftPhone:** A software-based phone emulator that allows a developer to call a voice application for testing. The Workshop Action class is used to start the SoftPhone application, which is an external component to WebLogic Workshop.
- **Log Viewer:** A developer may view the logs of all the different voice resources while the application is running or after the application has completed. The Log Viewer uses the Workshop Frames class to allow the log viewer to dock within Workshop and inherit the appropriate look-and-feel.
- **AppTuner:** A software-based speech platform for a laptop or PC that controls the speech recognition, prompt playing, and text-to-speech media components required for application testing.

Test Voice Application

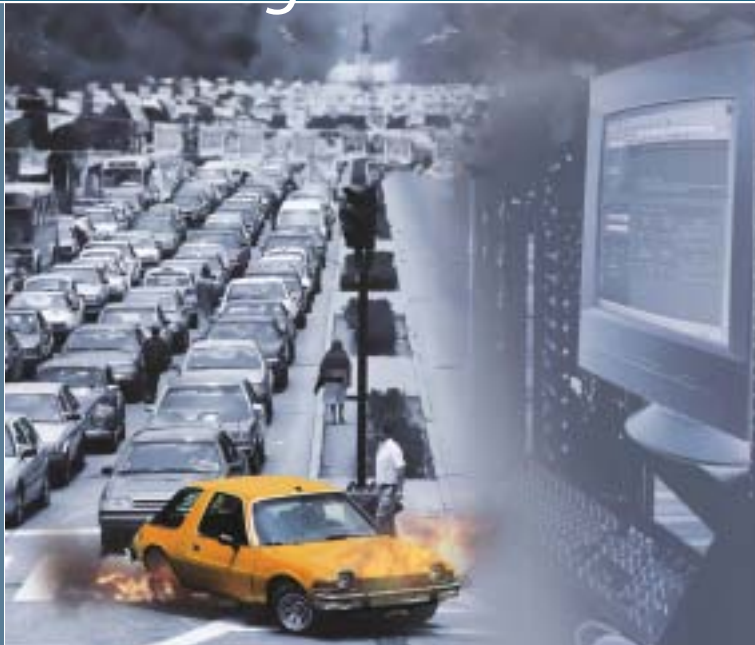
Test Voice Application simplifies the test process by allowing the developer to launch multiple components needed to test an application using a single button. Using the WebLogic Workshop Action class, this button starts the WebLogic Server to host the application, the Log Viewer to view speech resource logs, and the SoftPhone for calling the application. When launched, the SoftPhone is automatically populated to run the current VXML document.



What's Obstructing Your Application Flow?

With Cyanea/One™, your company will gain valuable insight into its J2EE applications. With a few mouse clicks, you can quickly drill down into your applications, all the way down to the method level. Isolate, diagnose and proactively resolve bottlenecks and resource consumption issues. Accomplish all these without touching your code or even requiring knowledge of your application.

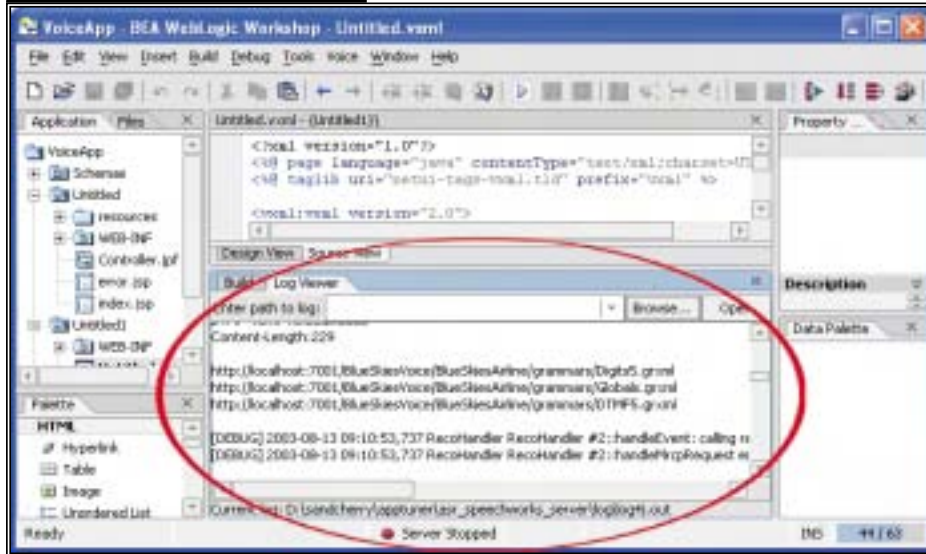
...*Finding* **Methods in the Madness.**



Find methods at:
www.cyanea.com/wldj/nomoremadness.html
1-877-CYANEA8 | sales@cyanea.com

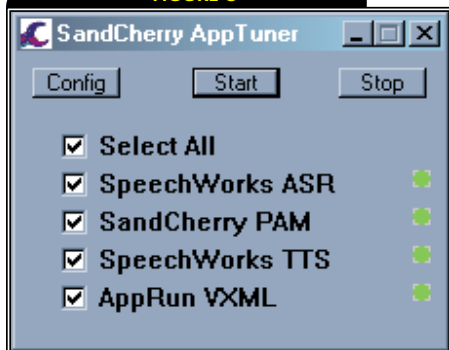
Cyanea™ and Cyanea/One™ are trademarks of Cyanea Systems Corp.
© 2003 Cyanea Systems Corp. All Rights Reserved.

FIGURE 7



Log Viewer

FIGURE 8



AppTuner view of speech component status

SoftPhone

The SoftPhone, like the audio document, is also a stand-alone program. Again, extending the Action class from the Voice Menu made it easy to start the SoftPhone without requiring any changes to the SoftPhone application. Other WebLogic Workshop extensions allowed SandCherry to retrieve current document information used to auto-populate the phone and specify which VXML document to run (see Figure 6).

Log Viewer

The Log Viewer provides access to important information regarding the status of the speech resources used to run a voice application. For example, as an application is processing speech recognition, the log viewer can be used to verify that the recognition engine is matching speech grammars correctly.

The Log Viewer is launched from the Voice Menu or toolbar via WebLogic

Workshop's Action class (see Voice Menu and Toolbar), seamlessly integrating into Workshop by using the Frame class. By default, the window has the appropriate look-and-feel of other WebLogic Workshop components and can dock into multiple locations (see Figure 7).

AppTuner

AppTuner, the underlying speech platform for testing voice applications, is a complex system with a number of integrated components. The AppTuner Console is a separate GUI program for starting and managing speech resource components, giving the application developer an easy to read view of the running speech resources (see Figure 8).

Originally, the console was going to be the only portion of the AppDev suite requiring start-up outside of the WebLogic Workshop window – unfortunate since all other voice development tools were embedded neatly within the Workshop framework. A simple solution exists within Workshop, however, that requires no code at all to add this to the Workshop environment.

WebLogic Workshop provides a tool for starting an executable from the Tools menu. Following the steps below made the stand-alone AppTuner Console an integrated tool within Workshop.

1. From the Tools menu, select IDE Properties.
2. In the left-hand pane of the IDE Properties Window, click on the Tools folder.
3. Select the New Tool button at the top of the right-hand pane.

4. In the Tool Name text box, enter “AppTuner Console”.
5. In the Directory text box, enter the full path to the AppTunerConsole.exe
6. Click OK.

The console is now available from the External Tools menu option within Workshop.

Summary

Integrating SandCherry's AppDev VXML into BEA WebLogic Workshop 8.1 provides a number of unique challenges due to the inherent differences between voice applications and Web applications. Incorporating a new project and file types for voice applications, dealing with files that contained only recorded audio, and incorporating external applications into Workshop were only a few of the potential roadblocks to offering a complete voice development environment seamlessly integrated into Workshop. Surprisingly, this effort was far easier than expected, despite moments that prompted head-scratching and calls to the BEA WebLogic Workshop team.

In the end, however, the extensibility components built into WebLogic Workshop 8.1 provided solutions for almost every challenge faced during integration. Extensible Project, Document, Frame, and Action classes offered tremendous flexibility for incorporating new capabilities into the WebLogic Workshop framework. In addition to supporting internal enhancements, Workshop's extensibility support provided different ways to incorporate external applications and even allow applications unrelated to the Web application server or its components to appear fully integrated. Developers can now build and test Web and voice applications seamlessly using BEA WebLogic Workshop 8.1 and SandCherry's AppDev VXML. 🍷

Listing 1

```
<action-set>
  <action
class="com.sandcherry.ide.bea.control.LaunchValidator"
    label="Validator" show-always="false"
    icon="resources/images/com/sandcherry/ide/Validator.gif">
    <location priority="30"
path="menu/voicemenu/voice" />
    <location priority="30" path="toolbar/voicetoolbar/voice" />
  </action>
</action-set>
```


As Java development evolves, so does JProbe®

JProbe®

Find the cause of J2EE code performance, memory and threading problems faster than ever before with Quest JProbe. New investigative features for finding memory problems combined with dramatic performance improvements mean even quicker resolution of problems in your application, servlet, JSP and EJB code.

JProbe Suite

JProbe Profiler

JProbe Memory Debugger

JProbe Threadalyzer

JProbe Coverage

Part of the **Quest Performance Management Suite for the J2EE Platform**



For more info and a free eval, visit:

<http://java.quest.com/jprobe/wldj>



Application Management with WebLogic Server for Developers

PART 2

APPLICATION DEPLOYMENT AND MONITORING TOOLS



BY VADIM ROSENBERG &
ROBERT PATRICK

AUTHOR BIOS...

Vadim Rosenberg is the product marketing manager for BEA WebLogic Server. Before joining BEA, Vadim spent 13 years in business software engineering, most recently at Compaq Computers (Tandem Division) developing a fault-tolerant and highly scalable J2EE framework.

Robert Patrick is a director of technology in BEA's Office of the CTO and coauthor of the book *Mastering BEA WebLogic Server: Best Practices for Building and Deploying J2EE Applications*. He has spent his career helping customers design, build, and deploy high performance, fault-tolerant, mission-critical distributed systems using BEA Tuxedo and BEA WebLogic Server.

CONTACT...

vadimr@bea.com
robert.patrick@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

Once your BEA WebLogic Server domain is configured, you need to deploy your application.

This is the second in a series of articles on BEA WebLogic Server administration and management for developers. The first article (*WLDJ*, Vol. 2, issue 10) introduced the major concepts and terminology for a WebLogic Server domain. Then, we showed you the most commonly used graphical tools for setting up WebLogic Server, and packaging and configuring applications to be deployed on it: the Administration Console, the Configuration Wizard, and WebLogic Builder.

In this article, we'll discuss application deployment, runtime management, and the monitoring facilities available with WebLogic Server – both graphical and command-line based.

Application Deployment with the Administration Console

In the deployment area of the Admin Console, you deploy new applications and modules – J2EE Enterprise applications, EJBs, Web applications, J2EE CA Connectors, and WebLogic Server Startup or Shutdown classes – to servers and clusters in the domain.

WebLogic Server enables you to deploy an application or module as either an archive file or an exploded archive file that contains the same files and maintains the same directory structure as the archive itself. Archived applications and modules must use the correct file extension for the module type:

- EJBs are packaged as .jar files.
- Web applications are packaged as .war files.
- Resource adapters are packaged as .rar files.
- Enterprise applications are packaged as .ear files.
- Web services are packaged as either .war or .ear files.

Exploded archives are frequently used in a development environment because they allow you to easily change parts of the application, recompile, and redeploy without regenerating the J2EE archive. To target an exploded archive, you select the top-level directory of the application or module, rather than the archive file.

After you have initially set up and deployed an application or module to one or more servers, you can stop, deploy, or redeploy the application or module without reconfiguring or re-copying its files, and without restarting the server. This is commonly referred to as hot deployment. Figure 1 shows an example of the Admin Console deployment screen.

When the server runs in development mode, it also supports something known as auto-deployment. With auto-deployment, the server watches the server's applications directory for new or updated files and automatically deploys or redeploys the new or updated archives.

Application and System Monitoring with the Admin Console

The BEA WebLogic Server Admin Console provides features for monitoring J2EE application components. The first and most obvious way to

make sure all components have deployed correctly and are available to the application is to find them in the JNDI tree. Figure 2 shows the JNDI tree displayed by the Admin Console.

Next, you can monitor the individual component usage as your application runs to confirm that the application is using the server facilities in the most efficient way. There are multiple places in the Admin Console that allow you to do this. For EJBs, the Admin Console provides extensive statistics about the EJB runtime usage that can help you determine how efficiently the server is able to process requests for each EJB. While the Admin Console provides a default view of a subset of these statistics, it also allows you to customize the view in real time to meet your demands for specific information. An example of a view into the EJB statistics is shown in Figure 3.

Similar to the EJB statistics, there is also a customizable display of servlets and JSP statistics, as shown in Figure 4.

You can also monitor many other J2EE and server-specific resources. Figure 5 shows an example of monitoring a JDBC connection pool while Figure 6 shows you how to monitor the general health of the server through the Admin Console.

Using the statistical and graphical information provided by the Admin Console, you can come up with initial estimates of your application's performance, see potential bottlenecks, and identify possible areas for optimization. While the Admin Console provides valuable application and server monitoring features, it may not provide enough information for all situations. In these cases, you have the following options:

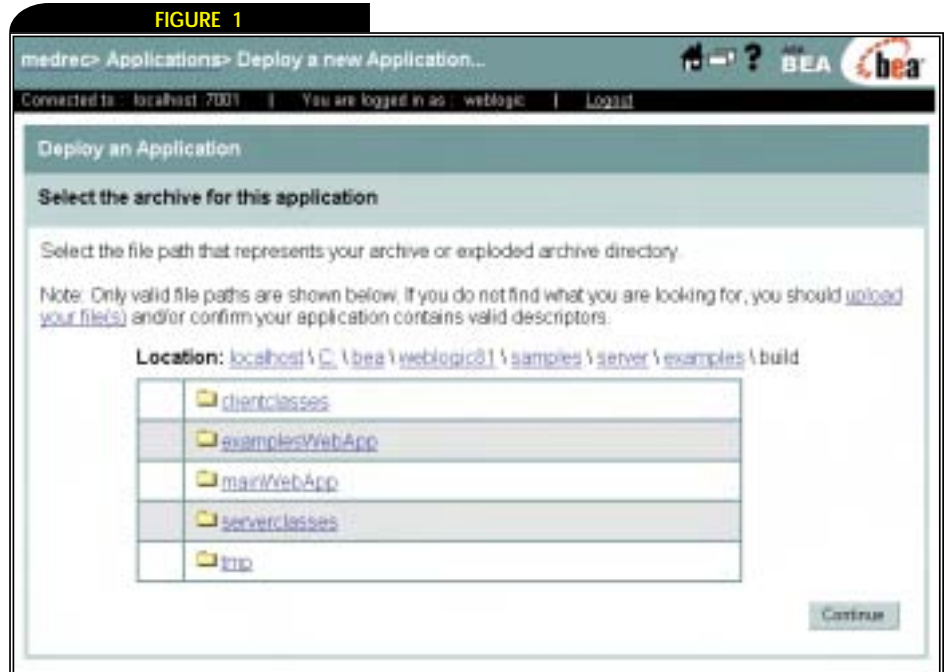
- Leverage BEA WebLogic Server's open and extensible management framework to add new monitoring features to the Admin Console, or access specific monitoring functionality from your custom applications.
- Leverage the extensive BEA partner ecosystem, which has numerous administration and management vendors who offer products that plug into WebLogic Server and provide more comprehensive administration, management, and monitoring functionality, in many cases covering the entire enterprise environment, not just applications deployed on WebLogic Server.

Command-Line Tools: `weblogic.Admin` and `weblogic.Deployer`

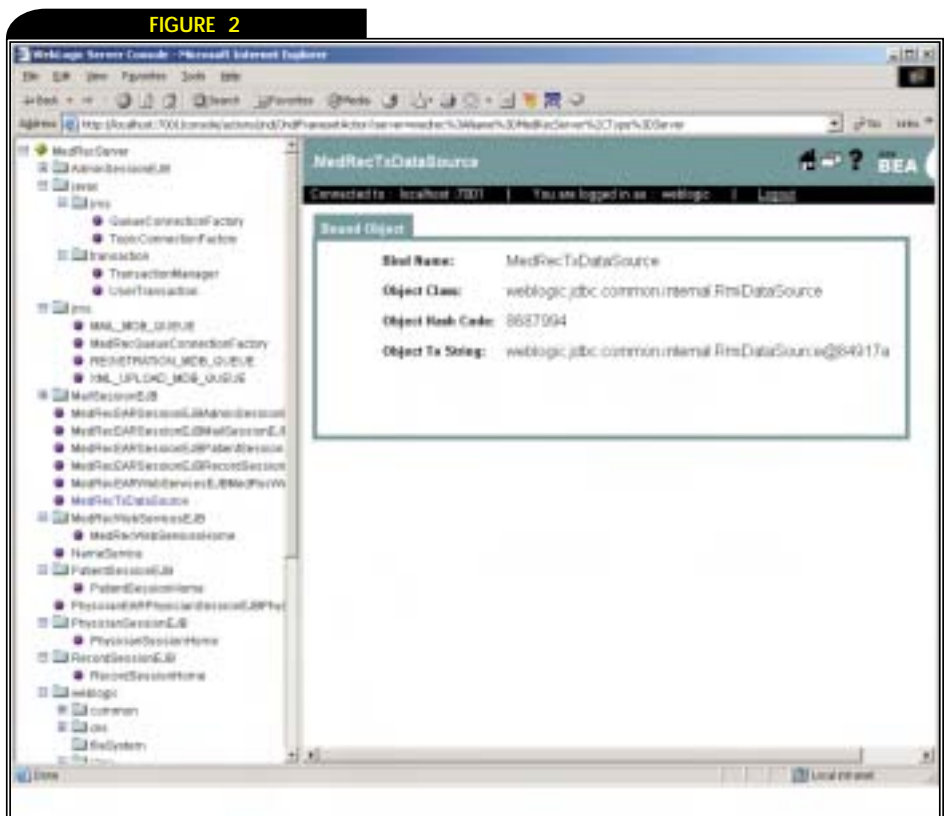
Graphical administration and management tools available with BEA WebLogic Server provide a powerful, intuitive, and

visual way to administer your WebLogic Server applications. However, in some cases simpler, more focused utilities are more convenient. In a development environment, a command-line interface provides a highly flexible way to automate

administrative tasks into your build, deploy, and test process. In a production environment, the command-line interface provides a repeatable process for accomplishing tasks to reduce the opportunities to introduce human error.



Application deployment screen in the Admin Console



Admin Console JNDI tree view

FIGURE 3

Entity Name	Access Local Count	Activation Count	Beans in the Context Count	Cache Access Count	Cache Size Ratio	Pool Max Ratio	Deployed Bean Ratio	Pool Element Ratio	Lock Obj. Waiter Ratio	Lock Obj. Timeout Ratio
GroupEJB	0	0	0	0	n/a	n/a	n/a	n/a	n/a	n/a
PatientEJB	8	2	2	4	833.33	080.00	080.00	080.00	n/a	n/a
VisitEJB	0	0	0	0	n/a	n/a	n/a	n/a	n/a	n/a
UserEJB	3	1	1	4	625.00	080.00	080.00	080.00	n/a	n/a
PhysicianEJB	0	0	0	0	n/a	n/a	n/a	n/a	n/a	n/a

A customized view of the Admin Console's Entity EJB statistics

FIGURE 4

Servlet Path	Execution Time Average	Execution Time High	Execution Time Low	Execution Time Total	Invocation Total Count	Pool Max Capacity	Request Total Count
/header.jsp	10	30	30	30	3	0	1
/viewRecords.jsp	391	391	391	391	1	0	1
/viewProfile.jsp	25	30	20	50	2	0	1

A customized view of the Admin Console's Servlet/JSP statistics

While not as visual and intuitive, the command-line interface does give you access to all of the functions available through the Admin Console, and even provides more flexible ways to access some features that are either hidden by the console or not that easy to use. Most of this access to administrative functionality is through the `weblogic.Admin` Java program's commands, which give you the ability to manipulate the underlying JMX infrastructure directly. To use these particular commands (GET, SET, CREATE, INVOKE, DELETE, and QUERY), you need to understand both JMX and BEA WebLogic Server's JMX MBeans. We will cover JMX in detail in a later article. The rest of this article will focus on the subset of commands that don't require an understanding of JMX, even though they are accessing the server's JMX infrastructure to accomplish their work.

The `weblogic.Admin` utility is a command-line interface that you can use to administer, configure, and monitor WebLogic Server. Like the Admin Console, this utility assumes the role of client that invokes administrative operations on the Admin Server, which is the central management point for all servers in a domain. While the Admin Console does everything through the Admin Server, the `weblogic.Admin` utility can access the individual managed server directly, as well as access the Admin Server. If the Admin Server is down, you can still use the `weblogic.Admin` utility to retrieve runtime information from managed servers and invoke some administrative commands. However, BEA WebLogic Server can only save configuration changes to the domain when the Admin Server is available.

Since `weblogic.Admin` is a Java program,

you need to set up your environment before you can run it. Once it's done, you can run it following this syntax:

```
java weblogic.Admin [ [-url | -adminurl] [protocol://listen-address:port]
                    -username username [-password password]
                    COMMAND-NAME arguments
```

If you want `weblogic.Admin` to use SSL to communicate with your WebLogic Servers, then you'll need to define some additional Java system properties to tell `weblogic.Admin` certain information. For example, the `weblogic.security.SSL.trustedCAKeyStore` property tells `weblogic.Admin` (and any other WebLogic SSL client or server) where to find the trust key store. Setting `weblogic.security.SSL.ignoreHostnameVerification` to true disables the normal process of verifying that the hostname in the server's certificate matches the server's IP address.

There is really no way for us to explain all of the commands available through the WebLogic Server command-line interface – there are just too many of them to fit here. For a comprehensive list of the available commands, see the WebLogic Server documentation at http://edocs.bea.com/wls/docs81/admin_ref/cli.html. We provide just a few examples here for you to get a feel for what you can do.

The `FORCESHUTDOWN` command instructs the Admin Server to shut down a managed server:

```
java weblogic.Admin -url t3://AdminHost:7001
                    -username weblogic -password weblogic
                    FORCESHUTDOWN MedRecManagedServer
```

After issuing this command, the BEA WebLogic Server instance named

`MedRecManagedServer` will shut itself down without waiting for in-flight requests to complete. When it receives this command, the server will print messages to its log file and to standard out, indicating that the server state is changing and that the shutdown sequence is starting. If the command succeeds, the final message that the target server prints is as follows:

```
<Jul 12, 2003 11:28:59 AM EDT> <Alert>
<WebLogicServer> <000219> <The shutdown sequence
has been initiated.>
```

In addition, if the command succeeds, the `weblogic.Admin` utility returns the following:

```
Server "MedRecManagedServer" was force shutdown
successfully ...
```

If the Admin Server is not available, you can send the `FORCESHUTDOWN` command directly to the managed server to have it shut itself down:

```
java weblogic.Admin -url t3://ManagedHost:7001
                    -username weblogic -password weblogic
                    FORCESHUTDOWN
```

If you want to write scripts that don't require the user to know an administrative username or password, you can tell `weblogic.Admin` to use the information stored in your boot identity file, also known as the `boot.properties` file. To do this, you need to tell `weblogic.Admin` where to find the boot identity file and where the root directory of your server is so that it can find the information it needs to decrypt the boot identity file (the default value is the current directory). The following example

Reports for BEA Workshop!



New! A BEA Workshop Control and toolset for embedding simple to sophisticated reports in every WebLogic Server or Portal application!

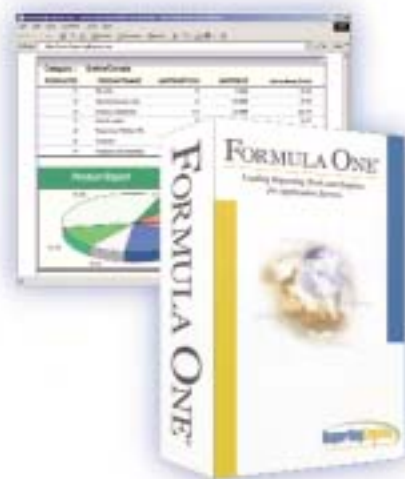
Design, preview, compile, and embed reports in your Java applications without leaving the Workshop environment.

Deploy directly to WebLogic Server or WebLogic Portal.

Point-and-click wizards help build reports from JDBC, Liquid Data, XML, and in-memory Java objects.

Output robust PDF, Excel, XML, HTML, DHTML, and email reports.

Formula One e.Report Engine Workshop Edition



Get Up and Running Quickly! Ideal for expert or novice Java developers. No separate report environments to learn. No separate report server to set up. No on-going report server maintenance.

Flexible! Use visual report creation tools or drive reports off user events with provided Java APIs and JSP tag library.

Cost Effective! Developer-focused pricing with unlimited deployments on your BEA server.

Leverage your BEA Platform! Inherits server features (scheduling, security, user administration, etc.) of your existing BEA WebLogic Server without duplication.



www.reportingengines.com
sales@reportingengines.com
888-884-8665 • 913-851-2200

Formula One e.Report Engine (Standard Edition)
Available Now! Get your FREE trial and training materials!
www.reportingengines.com/info/erengine1.jsp

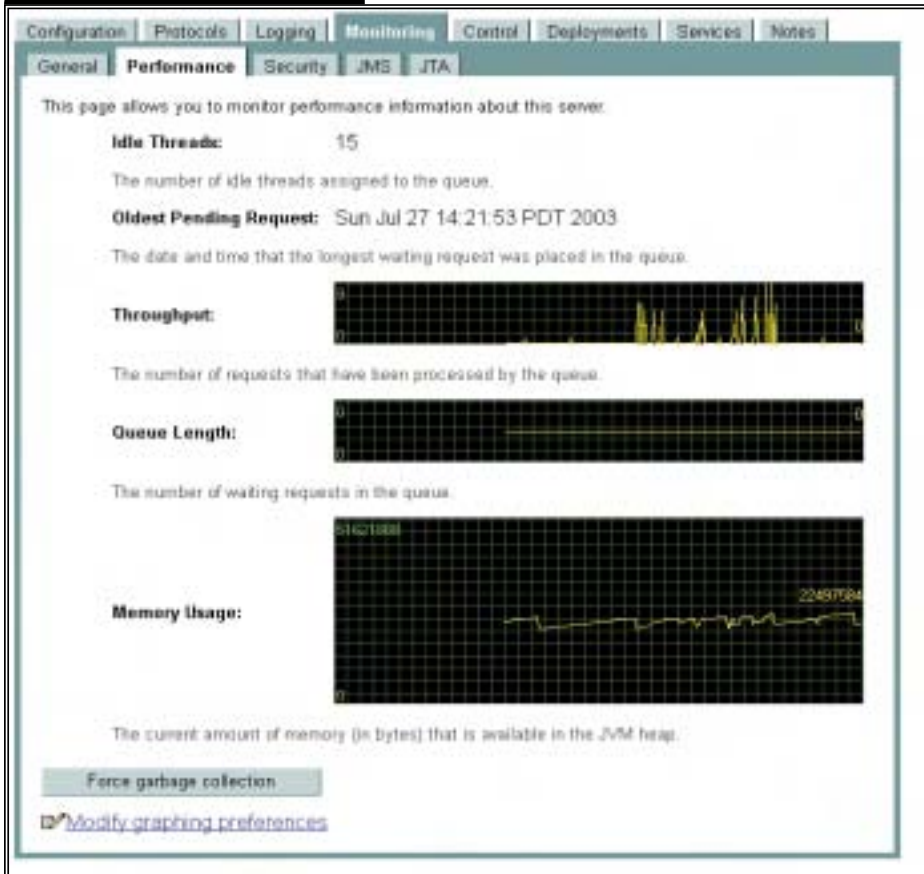
Formula One e.Report Engine Workshop Edition
Preview Release Available Now! Get your FREE trial!
www.reportingengines.com/products/bea/overview.jsp

FIGURE 5



Monitoring a JDBC connection pool

FIGURE 6



Monitoring the general health of the server

provides user credentials by referring to a boot identity file and specifies the server's root directory so that it can be invoked from any directory:

```
java Dweblogic.system.BootIdentityFile=
    c:\mydomain\boot.properties
-Dweblogic.RootDirectory=c:\mydomain
weblogic.Admin -url t3://AdminHost:7001
FORCESHUTDOWN
```

BEA WebLogic Server also provides a command-line utility for deploying applications called `weblogic.Deployer`. As with

`weblogic.Admin`, this utility is a Java program that requires certain arguments. The general syntax for `weblogic.Deployer` is:

```
java weblogic.Deployer [options] [actions]
[file(s)]
```

The most important options include the URL of the Admin Server, the username and password of a user with deployment privileges, the name of the application or component being deployed, and the list of targets to which to deploy the application or component. As you would expect, the

most important actions are deploy, redeploy, and undeploy. For a full list of the supported options and actions, please see the WebLogic Server documentation at <http://edocs.bea.com/wls/docs81/deployment/tools.html>.

To deploy the MedRec enterprise application archive to our managed server, you would use the command shown in Listing 1.

Just like `weblogic.Admin`, you can use Java system properties to specify SSL and/or boot identity file information (see Listing 2).

For a full list of BEA WebLogic Server command-line administration tools and their associated commands, please refer to WebLogic Server online documentation at <http://edocs.bea.com>.

Summary

This article showed you how to use the WebLogic Server Administration Console for managing BEA WebLogic Server and deploying and monitoring applications deployed on it. Then we introduced you to the command-line tools `weblogic.Admin` and `weblogic.Deployer` that provide the full scope of management functionality from the command-line interface.

In our next article we'll talk about JMX and how to use the tools and scripting facilities provided by BEA WebLogic Server for configuring, managing, and monitoring a WebLogic Server application. In the last installment of this series, we will dive into the Java APIs for building custom JMX programs, creating custom MBeans, and extending the Admin Console. 🍷

Listing 1

```
java weblogic.Deployer -adminurl
t3://AdminHost:7001
    -username weblogic password weblogic name
MedRec
    -targets ManagedServer1 deploy medrec.ear
```

Listing 2

```
java Dweblogic.security.SSL.trustedCAKeyStore=
c:\mydomain\trust_keystore.jks
Dweblogic.system.BootIdentityFile=
c:\mydomain\boot.properties
-Dweblogic.RootDirectory=c:\mydomain
weblogic.Deployer -adminurl
t3s://AdminHost:7002
    name MedRec -targets ManagedServer1 deploy
medrec.ear
```

The world, by way of Memphis.

FedEx, the very model of corporate efficiency, always looks for new ways to improve service. HP helped FedEx IT managers deploy HP OpenView,[™] which lets them identify and correct potential issues quickly and simply. The result is a smoothly running operation that produces happy customers from Memphis to Monaco — not to mention Mexico, Morocco and Martinique. www.hp.com/plus_fedex

fedex + hp

= everything is possible

FedEx



Automated WebLogic Server Domain Setup

SAVING TIME IN A COMPLEX ENVIRONMENT



BY ANDREAS WITTMANN

AUTHOR BIO...

Andreas Wittmann is a principal consultant with BEA Professional Services in Germany. He has worked in the area of TP Monitors and CORBA systems, and specialized in J2EE. Andreas has over 10 years of experience in OO technologies and distributed transaction processing.

CONTACT...

awittmann@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

BEA WebLogic Server domains in large-scale enterprises satisfy a broad range of requirements, including highly scalable application deployments, integration of various boundary systems, and high availability setups. As a natural consequence the level of the domain's complexity rises. Since similar WebLogic Server domains have to be created in various development, test, and production environments, we're looking for a solution that automates this process. The approach proposed here creates a clustered WebLogic Server domain in a multimachine environment, configures all services and resources, starts the domain, and deploys the applications in a fully automated manner.

Introduction

Large-scale enterprise applications are typically deployed to a number of environments, ranging from individual local developer environments to various test stages and integration and production environments. From a BEA WebLogic Server perspective, this means that we have to configure multiple domains on different machines. WebLogic Server administrators often have to quickly set up a fully functional domain,

starting from scratch, and handing over to some teams a running clustered WebLogic Server domain with connectivity to databases and other boundary systems configured, and various applications deployed. In highly complex environments, this can keep us busy for days.

For example, a customer's project domain for load testing was deployed to two 24-way machines in a cluster of 16 WebLogic Server instances, using 8 connection pools to 3 different databases and about 100 JMS queues. The target production release would increase the complexity of this environment by a factor of 5 - 10. A manual approach for this task is time consuming, error prone, and inflexible. We are looking for a one-step, script-based process to set up a full functional domain from scratch, or to rebuild an existing domain with different configuration settings.

Besides the initial installation of a WebLogic Server domain for a specific environment, there are many situations that will benefit from such a solution. For example, during an integration test, a domain was deployed in a directory that ran out of disk space. The alternative directory required that the domain be set up using a different Unix user account. A simple copy approach would probably break some links to deployment units and log directories and, due to tight security policies on that machine, would probably result in conflicting access rights.

Subsequent analysis and fixing activity can easily cost a couple of hours. Using the script-based approach we changed the installation directory and Unix user in a configuration file and rebuilt the domain at the new location. After 10 minutes the new WebLogic Server domain was started and ready to use.

Requirements

Before we jump into the details of this process, we will look at some requirements serving as guiding principals.

- **Fully automated process:** The process should run in a fully automated manner. No manual interaction should be required after the process is started. This process terminates successfully if the domain is set up, all managed servers are started, and all applications are successfully deployed.
- **Single configuration file:** There should be a single configuration file that controls the properties of the target domain. Since the process is split up into separate steps and each step uses various scripts and configuration files, this requirement guarantees the flexibility and applicability on different machines and for different domain configurations.
- **Platform independence:** This approach should be platform independent as far as possible. The first version was developed in a customer project on HP-UX, however; it is likely that the customer wants to extend the list of required platforms to include Windows and Linux.
- **Isolated domains:** The solution must provide for isolation of WebLogic Server domains. Different development and test teams will share a set of machines. Every team will need to control its WebLogic Server domain independently from other teams. As a result, no WebLogic Server resource should be shared across domains. Therefore, there must be separate node managers for every domain; otherwise changes in the node manager configuration will affect other domains.

We will see later in this article how these requirements are met by the proposed solution. Let's have a look at the process itself.

Process Overview

The automated process is composed of six steps. In the first step, we need a WebLogic Server domain in its simplest form, (e.g. a nonclustered single server domain consisting of an administration server only), and we need to start the Admin

Server of this domain. In the second step we extend this domain to a WebLogic Server cluster, managed by our Admin Server. We also need configurations for machines and the node managers. The third step will configure all required WebLogic Server services like JMS and JDBC resources as well as additional domain configuration settings. In a fourth step, we will copy the files of the application distribution to their target directories. In step five we start the node managers and the managed servers of the cluster. Finally, we will deploy and activate the applications. Figure 1 illustrates the six separate steps of this approach.

The BEA WebLogic product contains a number of tools that provide support during development and administration (i.e. WebLogic Workshop, WebLogic Builder, and the Domain Configuration Wizard). However none of these tools allows for a script-based automation for transforming an application built into a fully functional and running WebLogic Server domain.

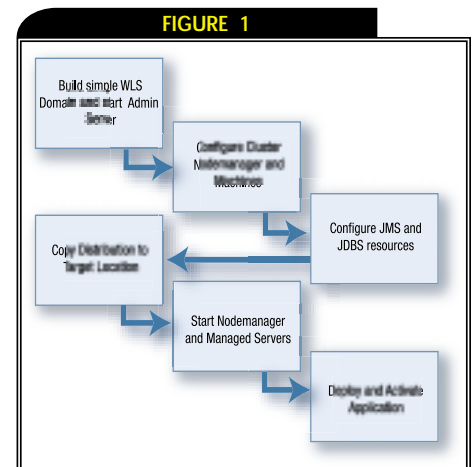
Therefore we combine several tools. The underlying tool suite is composed of the Apache Ant build tool, the BEA WebLogic Domain Configuration Wizard, the WebLogic node manager, and WLSshell, a JMX administration tool for WebLogic Server developed by Paco Gomez. Furthermore we make use of Unix shell scripts and SSH (Secure Shell).

We will use Ant as a basic frame for the automated process. Although we are not executing a build process in software development, there are a number of similarities where Ant seems to be the perfect choice. We want to be platform independent, we split up the whole process in separate steps which will become Ant targets and there are dependencies between these targets. Ant provides the flexibility to run the whole process in a single step or to execute individual targets separately. Furthermore there is a good integration of Ant and Java that we will use to start Java processes for the Domain Configuration Wizard and the WLSshell. Ant also provides capabilities to execute shell commands and wait for proper startup of HTTP services, which comes in handy when we need to verify that the Admin Server is started.

Figure 2 shows the complete automated process.

The basic frame is an Ant target that executes the whole process. Thus, we satisfy the requirement for a one-step process. We name the main target *rebuildAll* to highlight the fact that we also want to clean up everything before building a domain. This target is composed of subtar-

gets, which in turn can contain nested targets. This approach enables us to execute individual steps of the automated process separately, which is useful for error analysis as well as for managing an installed domain. We use three basic types of Ant targets. The first type uses Ant to work on the file system. We will use this target type to clean up a domain, to copy distribution files, and to build up the domain's directory structure. The second type executes shell scripts. This is a very convenient way to automate processes in Unix. The downside of shell scripting is its platform dependency. If we want to move to the Windows platform, we have to provide a Windows command file equivalent for every script. Furthermore, large scripts can become very complex and hard to read and maintain. The third type of Ant target calls Java programs, which are very well integrated into Ant. We use it in different situations (e.g. for executing the WLSshell). The WLSshell itself can read in nested WLSshell scripts.



The six steps comprising the fully automated process to build and start a clustered WebLogic Server domain

Template Approach

We want to have a single configuration file for changing domain-specific values, but we are using a number of different scripts and configuration files that need access to these values. The proposed solution is based upon the replace task from Ant. The only environment setting that we need is the path to the correct Ant executable.

We define variables with global relevance as properties in the file `domain.properties` that will be included by the `build.xml` file. If we want to use such a global variable in any other script or configuration file, we create a template file and

use a replacement token for this variable instead. A replacement token is any string that is enclosed in two "@" signs.

Template files reside in the `template.home` directory. We create an Ant target `trimFiles` to generate the target files from the templates and place them in the `project.home` directory.

Step 1: Simple WebLogic Server Domain

As part of an automated domain set-up, we need a script-based approach for setting up the initial domain. WebLogic Server 7.0 since SP1 features silent install of WebLogic Server where the setup of a WebLogic Server domain is a subactivity.

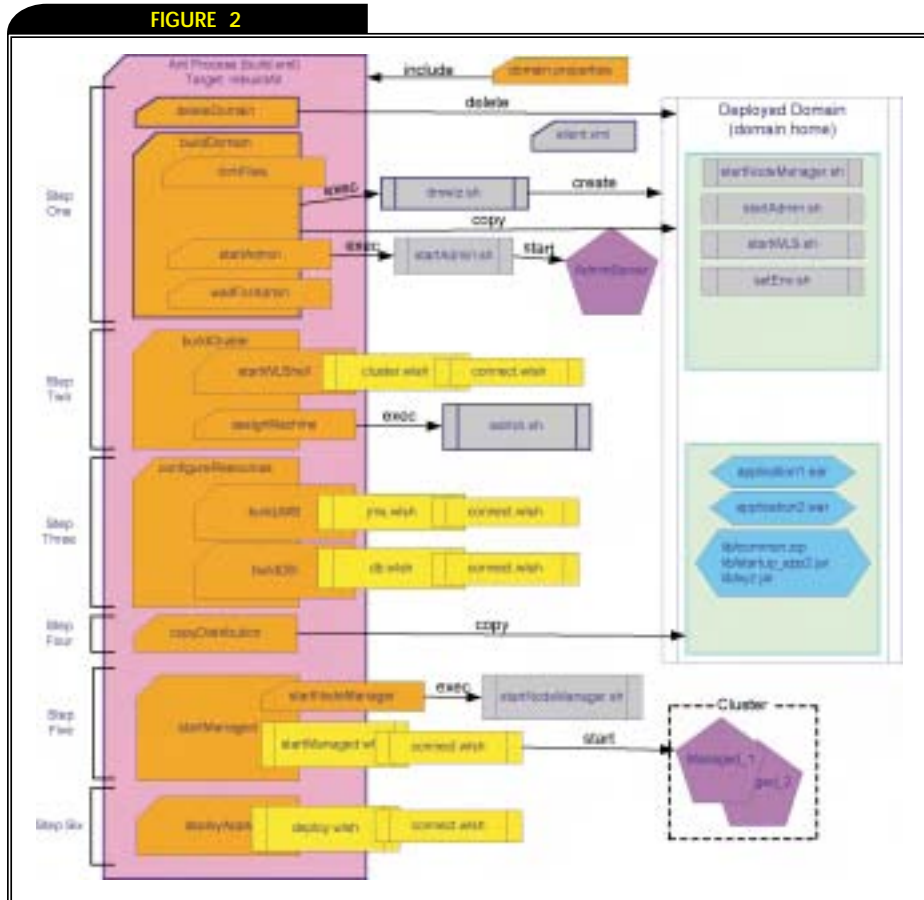
We automate the creation of a domain

with the Ant target `buildDomain`. This task requires the file `silent.xml` as input, containing the configuration information for the initial WebLogic Server domain. We only need the configuration for a very simple domain consisting of a single Admin Server. However, it must contain the correct values for the domain name, listen address and port, system user and password, and the correct path to the BEA Home and WebLogic Home. Since these values are domain specific we define them in the `domain.properties` file and use the template approach to generate the `silent.xml` file. The Ant target `buildDomain` calls the `dmwiz.sh` shell script, from the WebLogic distribution, which creates a new WebLogic Server domain from scratch in silent mode. Prior to this, it checks that all base directories exist and deletes any old domains at these locations. Since deleting open files can cause a lot of trouble on Unix systems, we verify that no process of that domain is running (e.g., we kill the node managers and all WebLogic Server instances). A successful execution of this Ant target will provide us with a newly created WebLogic Server domain that is ready to boot.

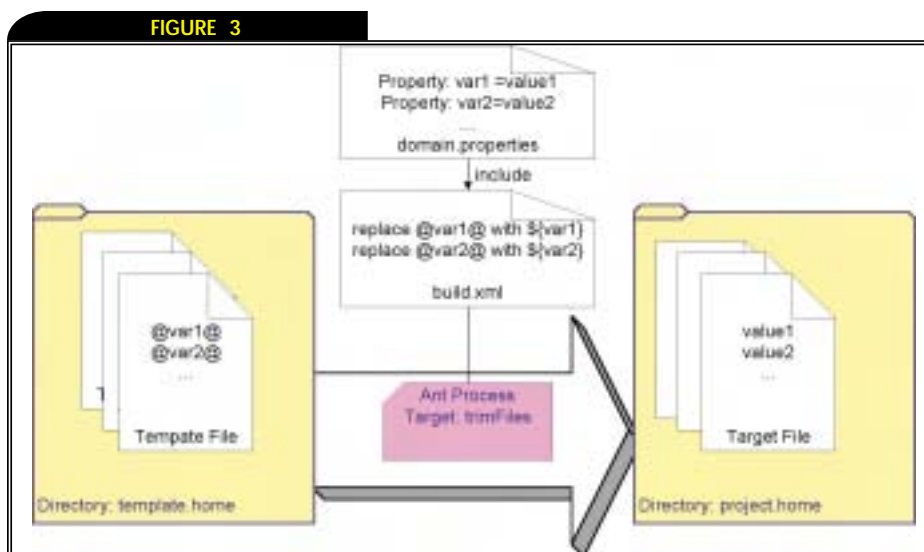
From now on, every configuration activity will be JMX based. Therefore we need to start the Admin Server of the new domain. We will use the Ant target `startAdmin` to call the shell script `startAdmin.sh`. This is a tailored version of the file `startWebLogic.sh` and contains some domain-specific information, i.e., domain name, listen address and port. Consequently we generate this script, using the template approach. Furthermore, we generate the file `boot.properties`, containing user and password, which prevents the Admin Server from prompting for this information. The Admin Server will be started with the "nohup" option; thus, it will continue to run even if we terminate its parent shell. But now we need to detect when to continue with the next step of the automated deployment. We have to wait until the Admin Server is running and accepting JMX commands. We implement the Ant target `waitForAdmin`, which realizes a very simple busy waiting loop, based on the Ant tasks `wait for` and `HTTP`. As soon as the Admin Server answers to a HTTP request, this loop terminates and we can continue with the JMX-based configuration.

STEP 2: CLUSTER CONFIGURATION

As a prerequisite for the cluster configuration, we need to have a WebLogic Server domain with a running Admin Server. The domain configuration is contained in the



The automated process based upon a framing Ant build file



Template approach using the Ant replacement target

Generate **dynamic** rich content in your
server-side Java applications

JClass[®] ServerViews

Add professional, dynamic charts and documents to your Servlet, J2EE and Web Services applications. Flexible and rock-solid, JClass ServerViews works with the latest JREs, web servers and application servers.

JClass ServerChart

Describe and pass charts as XML, load XML-formatted data, and output to browsers as Flash, GIF, JPEG and more.

JClass ServerReport

Generate high-quality Adobe PDF for a global audience.



Evaluate and experience JClass today – visit:
<http://java.quest.com/jcsv/wldj>

config.xml file. We could configure a domain by editing this file but that would require a complete domain restart, because changes to the domain configuration are saved by the Admin Server periodically but are loaded only at start time.

We prefer to use a JMX-based approach here and define server and cluster via JMX commands using WLSShell. Our working directory of the WLSShell is the project home. We name the script that configures the cluster `cluster.wslsh`. We would expect this script to contain domain-specific information inserted by the template approach. Instead, we use only variables here and read in a further WLSShell script, which we name `connect.wslsh`. `connect.wslsh` sets all domain-specific variables and connects to the Admin Server. It will be reused in all WLSShell tasks. Therefore `cluster.wslsh` is not a template script (e.g., following the template approach) while `connect.wslsh` is. `cluster.wslsh` defines a cluster, managed servers, and adds managed servers to the cluster.

In order to be able to start the domain via the node manager, we need to assign managed servers to machines. The script `cluster.wslsh` is the perfect place for this, since it defines the managed server and its attributes. However, using a bug of the WLSShell version prevents this. As a workaround, we use the WebLogic Admin Utility which gets called by the script `admin.sh`. The corresponding Ant target is `assignMachine`. Since all JMX commands can also be issued by the WebLogic Admin Utility, this approach presents a general workaround for all potential WLSShell bugs. But since the Admin Utility in BEA WebLogic Server 7.0 has no batch mode, and every JMX call has to be issued separately, this solution comes at the price of increased processing time.

Step 3: JMS and JDBC Configuration

The configuration of J2EE resources like JMS and JDBC is very straightforward. We use the Ant target `buildJMS` and `buildDB` here. Both Ant targets call WLSShell scripts, which in turn read in `connect.wslsh`. `connect.wslsh` establishes the connection to the Admin Server and defines all relevant variables for the JMS and JDBC configuration.

Since `connect.wslsh` is a template file, we can define all required variables in the `domain.properties` and feed them into `connect.wslsh` during the execution of the Ant target `trimFiles`.

Step 4: Copy Distribution

In this step we copy all application distribution files to the application directory of the domain home, thus making them available to BEA WebLogic Server. The distribution includes all EAR and WAR archives of the business application as well as additional libraries.

If we operate in a multimachine environment, we copy the whole domain (i.e., the domain root directory and all subdirectories) to the participating machines using FTP scripts or SCP (Secure Copy) in batch mode. We introduce a good level of redundancy, since only a subset of files is needed on the nodes that do not run the Admin Server. However, if the Admin Server node fails for some reason, we can quickly start the Admin Server on any of the other machines.

Step 5: Node Manager and Managed Server Start

In this step we want to start the managed servers of the domain. As a prerequisite we need to start the node manager on all involved machines. We use the Ant target `startNodeManager`, which calls a shell script `startNodeManager.sh`. This script must contain domain-specific settings (i.e., the listen port and the node manager home), so we use the template approach to generate it. For all remote machines of this BEA WebLogic Server domain, we use SSH commands for remote node manager starts.

Now we can start the managed servers using the Ant target `startManaged`. Since we want to start a domain with multiple managed servers we use the Ant task `parallel`, that provides fan-out capabilities and greatly reduces the startup time of a whole domain.

There was a customer project where we had to exclude some managed servers from the parallel server start, since they hosted a JMS service that was required by all other managed servers. Consequently, we fired off these managed servers first. We used the

script `startManaged.wslsh` to start the managed servers via JMX commands.

Step 6: Application Deployment

This is the final step of the automated domain setup. We deploy and activate the business applications on all nodes via JMX commands. We use the Ant target `deployApps`, which calls the WLSShell script `deploy.wslsh`. We do not have to wait until the managed servers have switched to running mode, because we are connected to the Admin Server, which is already running. The Admin Server starts a deployment task and takes care of the application deployment and activation.

Conclusion


The documented process has already proven it's usability during various customer projects.

Revisiting the requirements, we can state that the proposed solution satisfies "full automation", "single-place configuration", and "domain isolation". Platform independence is not fully satisfied because there are still various shell scripts, which are not directly supported on Windows platforms. However, projects can address this point by using Unix shell implementations for Windows, e.g. `cygwin`.

The presented solution was developed and tested with BEA WebLogic Server 7.0. WebLogic Server 8.1 introduces some interesting features that we might want to exploit to streamline this process. These features include a tighter integration with Ant, the Configuration Template Builder, and a batch mode of the Admin Utility.

The proposed approach to setting up a BEA WebLogic Server domain can easily be extended to automate some aspects of domain administration and management as well. In some customer projects, we included Ant targets to bounce individual servers or the whole domain. During load tests, we implemented Ant targets to change performance-relevant settings like JDBC connection pool sizes, number of worker threads, or JVM configurations for a whole cluster in a single step.

References

- *Ant Homepage:* <http://ant.apache.org>
- *WLSShell Homepage:* www.wlshell.com
- *WebLogic Server 7.0 Documentation:* <http://edocs.bea.com/wls/docs70/index.html>
- *WebLogic Server 8.1 Documentation:* <http://edocs.bea.com/wls/docs81/index.html>
- *Cygwin Homepage:* www.cygwin.com 

"Using the script-based approach we changed the installation directory and Unix user in a configuration file and rebuilt the domain at the new location"



©2003 Wily Technology, Inc. The Wily logo is a trademark of Wily Technology, Inc. Java is a trademark of Sun Microsystems in the U.S. and other countries.

It all comes down to you.

Your team. Your business. Your responsibility. Use a management solution that works for your whole organization. Wily Technology has been creating tools for managing Java applications longer than anyone else and nobody does it better.

Get Wily.™

Enterprise Java
Application Management
1 888 GET WILY
www.wilytech.com



Parallel
Business Logic
ProcessingParallel
Business Logic
ProcessingBY MURALI KASHABOINA &
BIN LIU

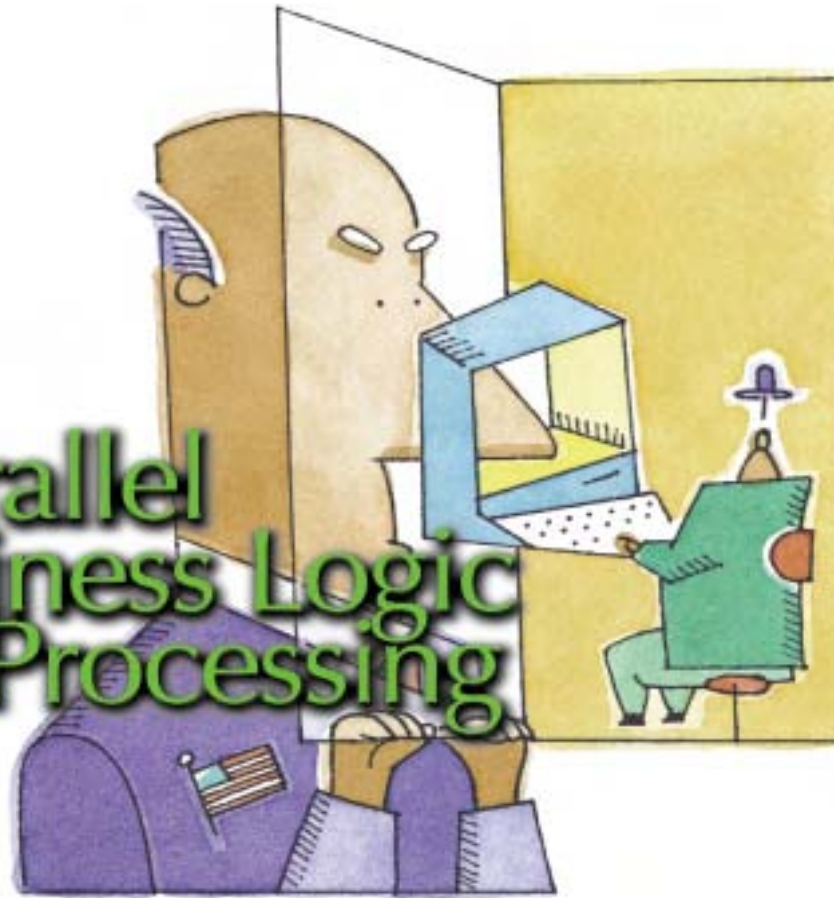
AUTHOR BIOS...

Murali Kashaboina is a lead software engineer at UAL Loyalty Services. He has more than six years of enterprise software development experience utilizing a broad range of technologies including J2EE, CORBA, and Tuxedo. Murali has published technical articles for SilverStream developer center.

Bin Liu is a senior software engineer at UAL Loyalty Services. He has more than five years of experience developing distributed applications using J2EE technologies, WebLogic, Tuxedo, and C++.

CONTACT...

murali.kashaboina@ualloyalty.com
bin.liu@ualloyalty.com

Parallel
Business Logic
Processing

A REUSABLE FRAMEWORK CAN SOLVE PROBLEMS

When independent business components must be executed simultaneously, the parallel processing of application business logic has a direct impact on the performance of the system; however, parallel processing at the application level historically has been challenging to implement.

In the case of J2EE applications that rely on the services provided by the application server containers, implementing parallel and concurrent processing is an uphill battle. Over time, the Java messaging services (JMS) API has become a de facto solution for parallel business logic processing by utilizing the asynchronous messaging features offered by the API.

Although JMS provides a solid foundation for developing J2EE applications that use asynchronous messaging, it does not automatically produce the desired end result. JMS poses the following challenges:

- JMS provides the pieces to solve the puzzle, but it does not put the pieces together to achieve an accurate solution.
- Even a JMS expert would have to overcome pitfalls and booby traps on the road to a successful implementation.
- Best practices disapprove of the scattering of complex JMS code throughout an application. Because of these challenges, a reusable, flexible,

and robust framework must be built to enable J2EE applications to exploit the asynchronous JMS features in order to achieve efficient parallel business logic processing. One such framework can be implemented by using JMS 1.0.2 and support for EJB 2.0 message-driven beans in the BEA WebLogic 6.1 Application Server.

A real-world example of this design strategy resulted in a reusable framework that can be used in any J2EE application development scenario that requires an asynchronous mode of execution. This particular instance employed a black-box strategy by utilizing traditional proxy and command patterns.

Typical Approach

When possible solutions to parallel business logic processing are considered, the first approach that comes to mind is spawning of threads. Multiple threads can be created to run each business task. This may work fine for simple applications; however, for enterprise applications that involve critical customer transactions and the exchange of huge chunks of data between multiple diverse systems, thread spawning may not be a robust solution. This type of design can lead to an interminable pursuit of perfecting the implementation itself because of the challenges involved in the management of thread life cycle, maintenance of the thread pool, management of transaction atomicity, and elimination of concurrency issues. An ounce of prevention is worth a pound of cure,

so eschew this approach in the first place and follow a robust design from the outset.

Design Summary

Multiple different strategies were considered for implementing such a framework. Certainly there could be myriad implementation approaches, but the best approach is the one that takes advantage of the underlying application server. One such approach was based on leveraging BEA WebLogic Server's high-performance JMS implementation.

The design strategy was based on sending messages containing action commands asynchronously to a JMS queue. Based on the implementation, an action command can perform any business task on execution. The main components in the design are:

- Action command
- Action message
- Facade bean
- JMS handler as a message producer
- Message-driven bean as a message consumer
- Message queue
- Temporary queue

The bean diagram visually depicts the participating components, the flow of request messages from the client of the framework to the message-driven bean, and the flow of response messages from the message-driven bean back to the client (see Figure 1).

Implementation Details

Action Command

The action command represents the actual business logic to be executed. A standard interface was created for the action command so that different implementations of the command can be plugged in based on different requirements. The command interface defines two important methods, shown in the following code snippet:

```
public interface Action extends Serializable{
```

```
    public void execute() throws Throwable;
    public Object getResult();
}
```

The action command is designed as a self-contained entity and is expected to contain the results of business logic execution. The getResult method returns the resultant object. This approach guarantees that the business logic and the results of business logic execution are completely isolated from the framework. The framework itself acts like a black box and only provides the means to execute the business logic, leaving intricate business logic details to the actual implementation of the action command.

Action Message

The action message acts as an adapter to the action command. The purpose of the action message is to shield the message-driven bean from dealing with the error conditions and the status of action command execution. Action message also helps the JMS handler correlate the messages sent to the message queue and the messages received from the temporary queue. The framework client can check for the status of the action command execution by inspecting the corresponding action message. The following code shows the important methods in the ActionMessage class:

```
public class ActionMessage implements
Serializable {
    .....
    public boolean hasThrowable() {
        return (throwable != null);
    }
    public void execute() {
        try {
            theAction.execute();
        } catch(Throwable t) {
            setThrowable(t);
        }
    }
}
```

Facade Bean

A facade was incorporated into the framework to prevent the client from knowing the intricate details of the JMS implementation. This facade is implemented as a stateless session bean. The facade bean is an entry point into the framework – the client of the framework creates a list of action commands and sends the list to the facade bean by invoking the sendAll method. The facade bean internally delegates the handling of action commands to a JMS message handler. It is responsible for looking up the JNDI context for the JMS message queue and connection factory and uses this information to create its own JMS message handler.

The timeout value on the sendAll method is used by the JMS message handler to set the timeout on the messages to be received from the temporary queue.

There are three main advantages to using the facade bean. First, it shields the clients from the intricacies of the JMS implementation. Second, the method invocation on the facade bean can take part in transaction management. Finally, because the EJB container pools the facade bean instances, the JMS resources such as temporary queues, queue sessions, and queue connections used internally by each bean instance will automatically get the pool effect, ensuring economical utilization of the resources. The facade session bean supports both remote and local invocations by defining remote and local interfaces (see Listing 1).

JMS Message Handler

The JMS message handler is the actual component that encapsulates the nitty-gritty details of JMS implementation. The AsyncMessenger class represents a JMS handler. The facade bean delegates the list of

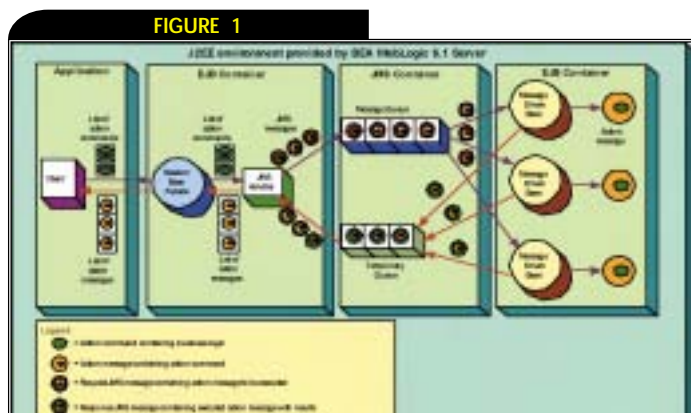


FIGURE 1 J2EE environment

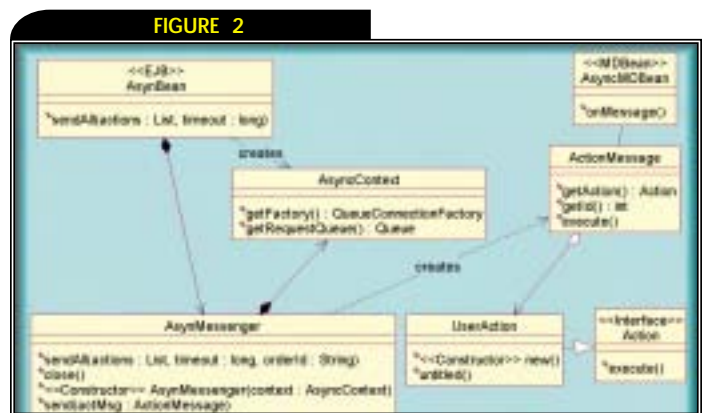
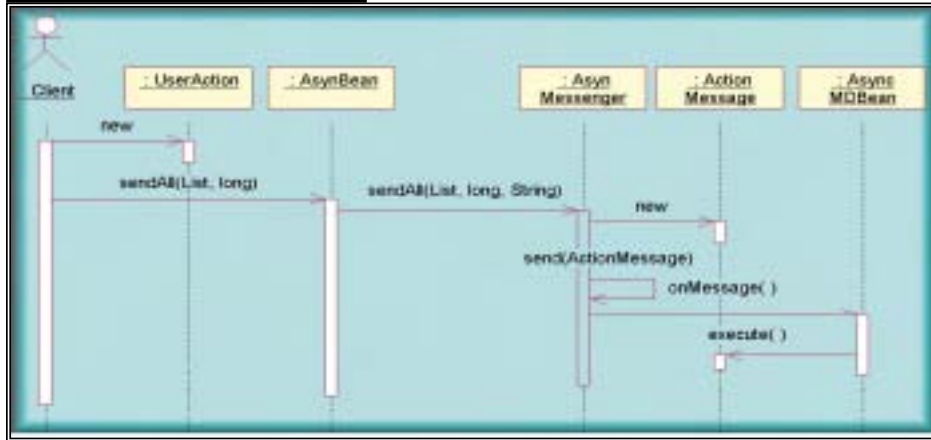


FIGURE 2 Class diagram

FIGURE 3



Class diagram

action commands to the AsyncMessenger instance by invoking the sendAll method. The AsyncMessenger instance first wraps each action command in an action message object. It then creates a JMS message instance of javax.jms.ObjectMessage type. The AsyncMessenger sets the action message object as a serializable payload in the JMS ObjectMessage instance. It also creates a temporary queue and registers a JMS queue receiver for receiving the response messages from the temporary queue. It includes a reference to the temporary queue in the JMS message. Finally, it sends the JMS message to the message queue asynchronously.

Since the execution time for each action command can vary, the sequence in which the response messages are received by the AsyncMessenger may not exactly match the

sequence in which the request messages were sent. In order to guarantee the exact mapping of the response message to the corresponding request message, a unique multiplexing ID is associated with each request message. This ID is nothing but the index of the original action command in the list. The AsyncMessenger sets this ID as an attribute in the request action message, retrieves it from the response action message, and uses it to set the action message in the response list at the same location as the original request message was in the input list.

Message Driven Bean (MDB)

An MDB is used as the message consumer to retrieve and process the messages transported via a message queue. The EJB container is responsible for managing the life cycle of MDBs. At runtime, the EJB container creates many MDB instances and keeps those instances in a pool. After creating an instance in the pool, the EJB container takes care of subscribing the instance to a specified message topic or connecting the instance to the specified message queue, based on the JMS information provided in the MDB deployment descriptor. The main advantage of using MDBs over custom JMS message consumers is that MDBs are inherently JMS message consumers and the EJB container provides a robust environment for MDBs. Because MDB instances in the pool can concurrently consume and process hundreds of messages, MDBs are capable of providing higher throughput and scalability than most traditional JMS message consumers.

In the current implementation, on notification of a JMS message, an MDB retrieves the action message from the JMS Object message and invokes the execute method on the action message. This invocation will trigger the actual execution of the action command contained by the action mes-

sage. After execution is completed, the MDB creates a new JMS Object message and sets the action message as a payload. It then retrieves the reference to the temporary queue from the input JMS message and sends the newly created JMS message to the temporary queue of the JMS handler. This point-to-point messaging assures guaranteed delivery of the response to the authentic original caller.

Message Queue

Point-to-point (P2P) messaging was chosen for the example implementation because it offers a guarantee of the message being processed once-and-only-once by one-and-only-one message consumer. In the example case, the business logic should be executed once and only once, but in tandem. The main component that makes P2P work is the message queue, which basically decouples the message producer from the message consumer. Several message consumers can connect to the same queue, but the underlying JMS server guarantees that the message in the queue is delivered to one and only one message consumer. In the example implementation, a message queue is configured as the primary destination for all the action messages (the MDB instances being the primary message consumers).

Normally, a message in the queue is delivered to the consumer in the order it was placed in the queue. The BEA WebLogic 6.1 JMS implementation provides the flexibility of sorting the delivery order of the queue messages by configuring destination keys. It enables "LIFO" and "FIFO" ordering of message queues. To enable FIFO (first-in-first-out retrieval of the messages in the queue), the sorting order of the JMS messages is configured by setting an ascending destination key in the BEA WebLogic 6.1 server JMS configuration.

Temporary Queue

A temporary queue is typically used by JMS clients to participate in request-response messaging using the JMS point-to-point protocol. A temporary queue belongs uniquely to the creating JMS queue session, and its life spans the entire duration of the queue session's connection unless the temporary queue is deleted. Unless its reference is transferred to other JMS clients using the JMSReplyTo header, the temporary queue is only accessible by the JMS client that created it. Although multiple different JMS clients can send messages to a temporary queue, JMS messages from the temporary queue can only be consumed by the JMS queue session

FIGURE 4



JMS connection factory

FIGURE 5



AsyncJMSQueue

associated with the JMS connection that created the temporary queue.

The BEA WebLogic 6.1 Server enables easy creation of temporary queues. A JMS template has to be configured on a BEA JMS server to enable the creation of temporary queues in that JMS server. A JMS template provides an efficient means of defining multiple destinations with similar attribute settings.

In the current framework, the JMS handler creates a temporary queue in order to receive responses from the MDB that executes the action messages.

Complete Picture

Figures 2 and 3 depict the relationship and message flow between the component classes.

Deployment Details

Deployment Descriptors

The first step in deployment is to create the deployment descriptors for the facade session bean and MDBs. The XML fragments shown in Listings 2 and 3 (due to space limitations, Listing 3 and 4 are online at www.sys-con.com/weblogsourcecfm). are a sample deployment description. The next step is to JAR up the EJBs and run

the BEA WebLogic ejbc utility. The final JAR file is ready to be deployed using the BEA WebLogic Admin Console.

BEA WebLogic 6.1 Setup

1. Configure the WebLogic JMS connection factory (see Figure 4).
 - a. In the left pane of the Admin Console, click on Connection Factory node and then click on the "Configure a new JMS Connection Factory" link in the right pane.
 - b. On the General tab, give the connection factory a name and a JNDI name. Fill in other attributes as appropriate, and then click on Create.
 - c. Fill in the Transactions and Flow Control tabs, as appropriate. Click on the Apply button on each tab when changes are completed.
 - d. On the Targets tab, target a WebLogic server instance or a server cluster on which to deploy the connection factory by selecting either Servers tab or Clusters tab.
2. Configure the WebLogic JMS template.
 - a. In the left pane of the Admin Console, click on the Templates node and then

click on the "Configure a new JMS Template" link in the right pane.

- b. On the General tab, specify a name for the template and click on the Create button.
 - c. Fill in the Thresholds & Quotas, Override, and Redelivery tabs, as appropriate. Click on the Apply button on each tab when changes are completed.
3. Configure the WebLogic destination key.
 - a. In the left pane of the Admin Console, click on the Destination Keys node and then click on the "Create a new JMS Destination Key" link in the right pane.
 - b. On the Configuration tab, specify a name, sorting property, key type, and sorting order direction.
 - c. After specifying all the attributes, click on the Apply button.
 4. Configure the WebLogic JMS Server.
 - a. In the left pane of the Admin Console, click on the Server node and then click on the "Configure a new JMS Server" link in the right pane.
 - b. On the General tab, specify the server name, a store if one has been created, a paging store if one has been created,

Web Services

Actional

MAKE SURE YOUR WEB SERVICES BEHAVE (AND PLAY WELL WITH OTHERS).

Yes, there *is* a way to ensure the manageability and interoperability of your Web services apps – and you can do it from within WebLogic, at dev time!

Introducing Actional for BEA, the market's leading Web services management technology, now purpose-built and immediately available for the BEA WebLogic Platform. Actional for BEA includes controls and tools that enable you to bring your Web services under management *the moment you develop them*.

Accelerate and simplify development, deployment, and management of your Web services apps. Find out how in a free technology guide created specifically for developer teams, architects, and IT managers. Download your FREE copy now and find out how to:

- Boost developer productivity
- Modify Web services apps more quickly and easily
- Ensure seamless, reliable operation of Web services applications
- Avoid the cost and complexity of unmanaged Web services
- And more!

Download your FREE guide now at www.actional.com/guide

FREE GUIDE
TO WEB SERVICES SUCCESS
DOWNLOAD NOW OR REQUEST CD-ROM AT
www.actional.com/guide



- and select the previously created template as a temporary template for the server.
- c. Click on the Create button to create the server.
 - d. Fill in the Thresholds & Quotas tab as appropriate, and then click on the Apply button to finalize the changes.
 - e. On the Targets tab, target a WebLogic Server instance and apply the changes.
5. Configure the WebLogic queue (see Figure 5).
 - a. Under the Servers node in the left pane of the Admin Console, expand the newly created JMS server instance and click on the Destinations node.
 - b. Click on the “Configure a new JMSQueue” link in the right pane.
 - c. On the General tab, specify the queue name and the queue JNDI name.
 - d. Choose the previously created destination key from the available list of destination keys and move it to the chosen list.
 - e. Click on the Create button to create the queue.
 - f. Fill in the Thresholds & Quotas, Override, and Redelivery tabs as appropriate; click on the Apply button on each tab to finalize the changes.
 6. Configure the facade session bean and MDBs.
 - a. Under the Deployments node on the left pane of Admin Console, click on the EJB node.
 - b. Click on the “Configure a new EJB link” on the right pane.
 - c. On the General tab in EJB component pane, specify the EJB name, path to the

- EJB jar file and deployment order.
- d. Click on the Deployed checkbox to set the deployment status for the component.
- e. Click on the Create button to create the component.
- f. On the Targets tab, target a WebLogic Server instance or cluster instance and apply the changes

Usage

Listing 4 shows a sample usage of the framework.

Benchmarking

Performance benchmarking was conducted with and without the framework on an airline e-commerce application that displays passenger itinerary information. The application retrieves data by interacting with many heterogeneous data sources, ranging from relational databases to mainframe legacy systems. Distinct data access objects are used to access data from different data sources and an aggregator service component was put into place to interact with a diverse set of data access objects and to assemble the retrieved data.

Without using the framework, the aggregator service would assemble data by invoking data access objects in a synchronous fashion. In this case, it was anticipated that the overall response time would be very high, and the application performance would be very poor. This expectation was proved accurate by conducting a load test on the application: if there are five itineraries on average per passenger, the average response time to retrieve complete information is about 35 seconds. The load test on the application proved that unless a parallel

and concurrent data retrieval mechanism is implemented, the optimization of response times will be exceptionally difficult.

The framework was incorporated into the application such that the aggregator service would simultaneously invoke data access objects asynchronously, and another round of load testing was conducted on the application. This test yielded positive results indicating an average response time of about seven seconds. Thus, application performance increased almost five fold by using the framework.

Conclusion

A reusable, extensible, and robust framework that utilizes the asynchronous features offered by JMS and the robust runtime environment provided by the EJB container can solve the problem of parallel and concurrent business logic processing in enterprise and mission-critical applications. The benchmarking of an airline application running in a BEA WebLogic 6.1 Application Server yielded very positive response times and higher performance when the framework was incorporated into the application for parallel and concurrent data access from multiple data sources. The implementation of the framework proved to be much more robust than the typical thread-based implementation.

Acknowledgments

We would like to thank Virgil Bistriceanu, managing director, and Helen Agulnik, manager, for their support and inspiration in writing this article. We would also like to extend our special thanks to our colleague Kelly Oliver, who contributed her time by reviewing and editing the article. 🍷

Listing 1: Facade session bean

```
public interface Async extends javax.ejb.EJBObject {
    public List sendAll(List actionList, long timeout) throws
    RemoteException;
    public List sendAll(List actionList, long timeout, String orderId)
    throws RemoteException;
}
public interface AsyncLocal extends javax.ejb.EJBLocalObject {
    public List sendAll(List actionList, long timeout);
    public List sendAll(List actionList, long timeout, String orderId);
}
```

Listing 2: ejb-jar.xml

```
<enterprise-beans>
  <session>
    <display-name>AsyncBean Ejb</display-name>
    <ejb-name>AsyncBean</ejb-name>
    <home>com.uls.common.dao.async.AsyncHome</home>
    <remote>com.uls.common.dao.async.Async</remote>
    <local-home>com.uls.common.dao.async.AsyncLocalHome</local-home>
  </session>
  <local>com.uls.common.dao.async.AsyncLocal</local>
  <ejb-class>com.uls.common.dao.async.AsyncBean</ejb-class>
  <session-type>Stateless</session-type>
  <transaction-type>Container</transaction-type>
  <resource-ref>
    <description />
    <res-ref-name>async/ConnectionFactory</res-ref-name>
```

```
<res-type>javax.jms.QueueConnectionFactory</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
  <resource-env-ref>
    <description />
    <resource-env-ref-name>async/RequestQueue</resource-env-ref-name>
    <resource-env-ref-type>javax.jms.Queue</resource-env-ref-type>
  </resource-env-ref>
</session>
<message-driven>
  <display-name>AsyncMDBean</display-name>
  <ejb-name>AsyncMDBean</ejb-name>
  <ejb-class>com.uls.common.dao.async.AsyncMDBean</ejb-class>
  <transaction-type>Container</transaction-type>
  <message-driven-destination>
    <destination-type>javax.jms.Queue</destination-type>
  </message-driven-destination>
  <resource-ref>
    <description />
    <res-ref-name>async/ConnectionFactory</res-ref-name>
    <res-type>javax.jms.QueueConnectionFactory</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
</message-driven>
</enterprise-beans>
```

International Conference & Expo

Edge 2004 EAST

Development Technologies Exchange

February 24-26, 2004

Hynes Convention Center, Boston, MA

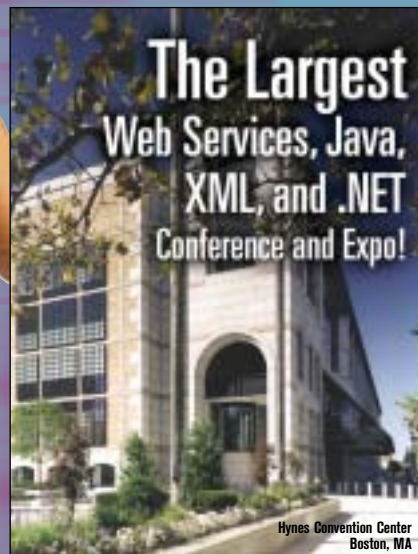
**ARCHITECTING JAVA, .NET, WEB SERVICES,
OPEN SOURCE, AND XML**

Addressing:

- ▶ Application Integration
- ▶ Desktop Java
- ▶ Mobility
- ▶ ASP.NET
- ▶ VS.NET
- ▶ JavaServer Faces
- ▶ SOA
- ▶ Interoperability
- ▶ Java Gaming

Register By
November 21, 2003

Up To
SAVE \$400



For more information visit
www.sys-con.com
or call
201 802-3069

Over 200 participating companies will display and demonstrate over 500 developer products and solutions.

Over 3,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 100 of the latest sessions on training, certifications, seminars, case-studies, and panel discussions promise to deliver real world benefits, the industry pulse and proven strategies.

Full Day Tutorials Targeting

- Java • .NET • XML
- Web Services • Open Source

Conference program available online!
www.sys-con.com/edgeeast2004

Contact information: 201 802-3069 • events@sys-con.com





TRANSACTION MANAGEMENT

This month's article is again inspired by an interesting design discussion posted on the weblogic.developer.transaction_newsgroup. (Ever get the feeling I'm running short of inspiration? Ideas for new articles always welcome!)

the transaction is committed, so why can't the logic in the MDB see the new entity bean? The transaction manager is broken, right?

Well, no. In order to understand this situation, you need to take a step back and think about the implementation the transaction manager does. From the 10,000 ft level, things should be working: the devil must be in the detail... Let's go diving!

Let's Dive for the Devil

A transaction encompasses the entity creation and the sending of the JMS message, so they will complete as an atomic unit – either message sent and entity created or total failure, that's what the transaction manager is giving us. However, from an implementation perspective, we need to look more closely at exactly when the transaction is complete. It can't be when the application (or the EJB container) calls commit – we know that this just initiates a set of dialogues between the transaction manager and the resource managers, which is bound to take some time. The completion will happen some time later, when these dialogues are done. Diving even deeper, you may recall that these dialogues fall into two categories – the two phases of the transaction (it's called two-phase commit, after all) looking at the xa specification. You'll find that once a resource manager has replied affirmatively to a prepare, it is undertaking to guarantee to make whatever updates were in the scope of the transaction at some time in the future. Now we're getting somewhere – we've found a period of time over which things will be happening behind the scenes; maybe these asynchronous things are causing our problem. From a high-level perspective, given the xa guarantee, the transaction can be assumed to be complete once the prepare calls have all succeeded. From an implementation level, until the commit calls are processed by all the resource managers we cannot be certain that we will be able to access the updated database state, and we have no way of knowing exactly when these commits will happen – commit processing is going on in the background and the time it takes to perform a commit will vary depending on factors such as system load, resource-manager locality, the order the transaction manager sent out commit messages in, and so on. (This ignores completely the possibility of failure; imagine the database manager crashing after a prepare. The commit can't be processed until it is brought back online. How long will that take? Well, it depends on how long it takes to fix the problem – if the crash is caused by a faulty

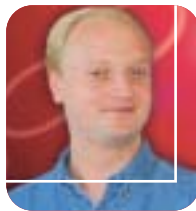
Transactions: How Big Are Your Atoms?

A STICKY PROBLEM THAT'S NOT ALL THAT UNCOMMON

BY PETER HOLDITCH

Since the problem described is a common one with transactional design I thought it might be valuable to review the design, the problems with it, and some solutions.

The problem was stated on the newsgroup thus:



We have a Session Bean method (with a "Required" TX attribute) that creates an entity Bean, and then fires a JMS message that indicates that it was created. There is an MDB that listens for this message. When it hears it, it looks up the entity bean.

The problem is that sometimes the lookup of this entity bean will throw an ObjectNotFoundException. We have ensured that the JMS message firing uses the transaction context of the method, so that the creation of the entity bean and the firing of the message all takes place within the same transaction (we did this by using the "javax.jms.TopicConnectionFactory", and using a JMS session that was not transacted). Also, we have verified that the entity that gets created exists in the database (at least it does sometime after the lookup by the MDB fails).

So, what's going on here? The creation of the entity bean and the sending of the JMS message are in the same transaction, and we know therefore that the message will not be dispatched until

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

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.



power supply in a machine, then it could take days waiting for a spare part. This whole parenthetic discussion then leads into one of my favorite subjects, the transaction abandonment timeout.)

So, the moral of this story is that you cannot rely on an atomic transaction being truly atomic in time – it will complete as a logically atomic unit, sure, but there will always be amounts of timing jitter involved in making its results visible across all the resources it touched.

Danger: Mixed Synchronicity!

It is clear now what the problem is with the design stated on the newsgroup. The assumption has been made that this asynchronous transaction processing doesn't happen. A race has been set up between the JMS and the database resource managers to commit the transaction. When JMS wins, the message-processing logic assumes the database has committed too, but it hasn't – the commit processing is still going on in the background, and the ObjectNotFound exception is thrown.

So much for the theory, how can we fix the design? There are (as always in architecture of this kind) a few options, ranging from the hacky workaround to the elegant rearchitecture.

The hacky workarounds involve coding around the problem, either with JMS message birth times or defensive coding in the MDB. If the code that creates the JMS message sets the birth time for some time in the future, the JMS system will introduce a delay into the processing path before it releases the message. This delay should give enough time for the commit processing to complete. That's a great theory as far as it goes, but how long should the delay be? As I already said, the required window will depend on system load and physical architecture, and it might vary radically in some failure conditions. Using this method for a production system will sooner or later lead to sporadic failures as loads and deployment vary, and will incur a support cost and a reliability loss. So, the defensive coding. A simple-minded approach might be to roll back the MDB, have the message redelivered, and try again; or simply try again after a pause in the MDB logic itself. That's well and good, but what if the scenario isn't object creation, but modification? Now you can't be certain that the data you're updating is the current data (at least, you'll not be certain that you're certain – it depends on

the database's locking strategy); to code around this, you add a version field to the object and implement some kind of optimistic concurrency so that the MDB can wait until it's sure it's operating on the right version of the data.

The fact that you're doing all this frantic coding to work around this issue should be ringing alarm bells – clearly the architecture of the application does not mesh well with the architecture of the infrastructure. The best solution is to get to the bottom of why...

It's Not a Mesh, It's a Mess!

JMS is all about allowing processes to run asynchronously with respect to one another. JTA is all about making updates that logically execute atomically, which in turn implies synchronously (or as near synchronously as reality allows). In this scenario, an attempt is being made to use JMS as a synchronous calling mechanism – the operations on the data are clearly related to one another (synchronous) but for some reason we have interposed an asynchronous messaging system into the processing flow. Maybe the most elegant solution would be to implement the next processing step as an Entity EJB, call it via RMI, and have it participate in the original transaction. All the updates would be visible to all the processing steps then, since the updated data is visible before the commit within the transaction. But what if there's another requirement that necessitates the asynchronous path to the "stage 2 processing"? Well, wrap the Entity EJB you created in this use case in an MDB facade and the logic can then be executed synchronously or asynchronously, depending on the use case (even better, maybe the "stage 2 Entity" only offers a local interface).

As a parting observation, this kind of tricky asynchronous corner case is not at all uncommon in building transactional systems – in fact, it's more like the norm. TP systems like Tuxedo, CICS, and others all offer facilities analogous to the design pattern I just described to handle this kind of thing. So does the BEA WebLogic Workshop framework – it builds in this style atop J2EE and provides a natural, event-driven programming model while taking care of this kind of implementation detail in the framework, again demonstrating the power and potential of using such a framework to simplify implementation while increasing reliability. ●

Once you're in it...



...reprint it!

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

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

Reprints



How to Secure a Web Application

WEBLOGIC SERVER EXTENSIONS CAN DOVETAIL WITH J2EE

While security is a concern throughout an application, it is especially important for Web application components. An insecure Web application leaves a Web site vulnerable to many attacks, some that require nothing more than an Internet browser and a small amount of knowledge.



BY NEIL SMITHLINE

AUTHOR BIO

Since beginning Java programming in 1996, Neil has implemented and architected several J2EE applications. His current role as BEA WebLogic Server security architect exposes him to all aspects of J2EE security. Neil has a bachelor's degree in computer science from SUNY Buffalo and a master's degree in computer science from the University of Rochester.

CONTACT...

neil.smithline@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

missions granted to users, allow a site to permit certain users access to different parts of the site. There is a declarative means of restricting specific URLs to users who belong to a set of roles, as well as a programmatic interface to determine if the current user is in a specific role.

Sample Web Application

Throughout this article, a sample application is developed. A skeleton of the application is available at www.sys-con.com/weblogic/sourcec.cfm. The skeleton contains all of the Web pages and deployment descriptors for the Web application but omits all content not directly pertinent to security.

The application has been tested on BEA WebLogic Server 8.1. Besides deploying the Web application, two things must be done to enable it to run:

- You must create a "Customers" group. This can be done under the Security >Realms >my-realm>Groups tab in the console. There is no need to populate the group, it just needs to be created. The application will populate it as needed.
- You must create a user named "weblogic" and put it in the "Administrators" group. This can be done under the Security >Realm>myrealm >Users tab in the console.

Why these actions are required will be explained as they arise.

Encryption (SSL)

In order to guarantee confidentiality while interacting with users, sites generally use SSL to perform encryption of sensitive pages. In the browser, SSL is signified by a URL beginning with "https" instead of "http".

The global and anonymous nature of the Internet means that a Web application can be attacked from anywhere in the world by individuals who frequently cannot be traced. Even if they are found, international law often makes prosecution all but impossible.

Java 2 Enterprise Edition (J2EE) provides many security features. Understanding when and how to use these features is complex and error prone. Furthermore, J2EE security falls short of satisfying many of the needs of Web application developers. BEA WebLogic Server implements all of the J2EE Web application security features and extends them to help complete some tasks that are impossible using J2EE APIs alone.

Web Application Security

J2EE provides two basic mechanisms for securing a Web application.

- **Encryption:** There is a mechanism for specifying whether SSL (secure sockets layer) must be used when accessing a specific URL. When SSL is used, it ensures that the communication between the user's browser and the server can neither be viewed nor modified by intermediaries. SSL also allows users to verify that they are sending the (presumably confidential) information to the intended destination and not to an imposter.
- **Role-based security:** Roles, a collection of per-

Unfortunately, as is the case with many security technologies, using encryption adversely affects performance. For this reason, SSL is typically enabled only for sensitive pages. A sensitive page is any page that contains confidential data, either outbound from the server or inbound from the user's browser via a form. There are a few exceptions to this rule.

Non-sensitive pages that are in the middle of an otherwise secure conversation should generally be encrypted (e.g., an advertisement in the middle of a purchasing transaction). Besides being convenient for the developer, many browsers notify users with a prompt when the browser changes from an encrypted conversation to a clear-text conversation. To avoid these potentially confusing prompts, use encryption for these nonsecure pages.

Also, pages that contain forms should be encrypted if the data to be filled in by the user will contain sensitive information. For example, consider a form that requires you to enter your name and credit card number. When the form is submitted it must use encryption to protect the confidential information, but the form itself, even if it does not contain confidential information, should be encrypted as well. This is needed because browsers generally display a lock in their bottom margin when a page has been encrypted. Some users will not enter secure data into a form if they don't see the lock. Note that this lock is irrelevant to whether the form data, the information the user actually wishes to secure, will be encrypted. The lock only signifies that the form has been encrypted. So, despite the lock usually being irrelevant, encryption should be used to ease users' concerns.

You can require a page to be encrypted by specifying a transport guarantee of either

"INTEGRAL" or "CONFIDENTIAL" (see http://edocs.bea.com/wls/docs81/webapp/web_xml.html#1019727 for details).

For a concrete example, consider a very simple Web application that allows online shopping. It contains the following files:

- **catalog.jsp:** Allows browsing and searching of the catalog. Items can be added to the shopping cart from the catalog.
- **shoppingcart.jsp:** Displays contents and allows manipulation of the shopping cart. Users can either purchase the cart's contents or shop further.
- **billinginfo.jsp:** Gathers shipping address and billing information and POSTs it to orderstatus.jsp.
- **orderstatus.jsp:** Executes the order. If successful, it displays a success message; if not, it returns to billinginfo to display the error and allow the user to correct it.

Figure 1 shows the flow for this site as well as the required use of encryption. Only the orderstatus.jsp page requires the use of encryption, as the post of the form from billinginfo.jsp contains sensitive data. That being said, it is recommended that billinginfo.jsp be encrypted as well so that users see the lock on their browser's toolbar. Listing 1 shows a fragment of a web.xml that would be used to establish these encryption requirements.

When you specify a <transport-guarantee> that requires encryption, BEA WebLogic Server will not allow a non-SSL connection to occur. If a non-SSL connection is attempted, WebLogic Server will send a redirection to the browser to cause it to access the page via SSL. This provides a convenient means of converting to SSL. That is, shoppingcart.jsp does not need to specify that SSL should be used for billinginfo.jsp. The application server will auto-

matically ensure this. For example, in our example Web application, the link in shoppingcart.jsp to go to billinginfo.jsp is:

```
<a href="billinginfo.jsp"> Checkout </a>
```

It does not specify HTTPS. WebLogic Server will ensure that HTTPS is being used.

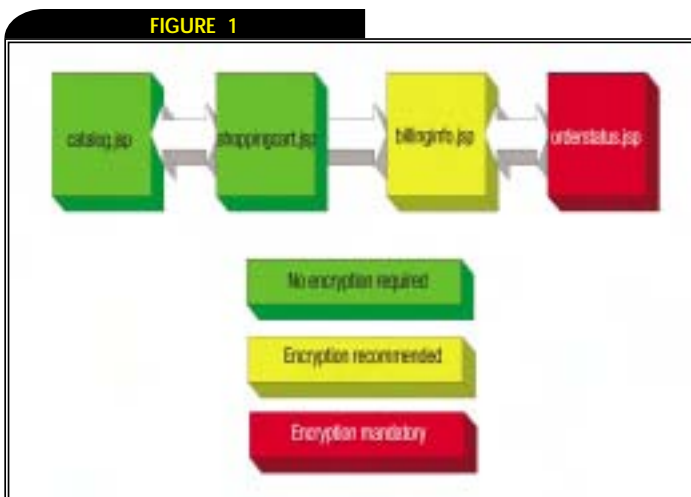
Two issues that arise from this behavior are:

1. You should never rely on this automatic conversion from SSL for secure form data. Once the data is sent to the server unencrypted, it doesn't matter if it is re-sent encrypted. By making the form page encrypted, as the sample Web application does, it ensures that this problem doesn't occur. The request for the form may be sent twice, first unencrypted and then encrypted, but the secure information in the form data will only be sent encrypted.
2. There is no way to automatically go back to non-encrypted communication. If you never specify a protocol, once SSL is selected it will be used for the remainder of the interaction. In our example, all links back to the catalog explicitly specify HTTP. Due to its frequent use and large amount of data (especially pictures), encrypting the catalog would be a performance nightmare. In our example Web application, we explicitly specify HTTP whenever linking to the catalog, as in the following:

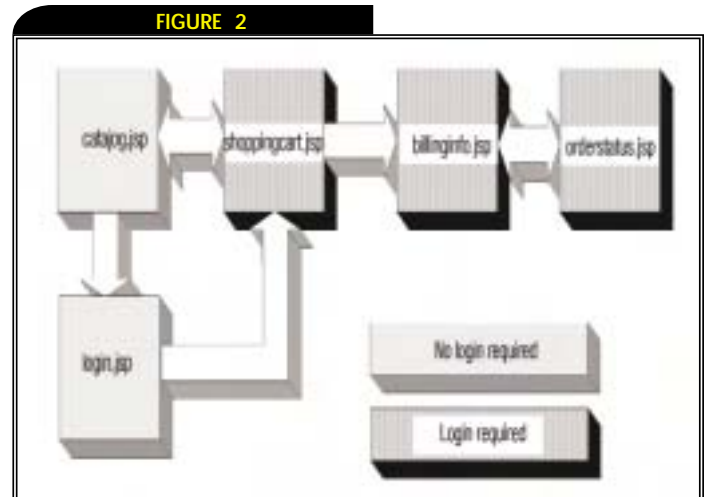
```
<a href="http://<%=request.getServerName()%>:<%=request.getServerPort()%><%=request.getContextPath()%>/catalog.jsp"> Continue Shopping </a>
```

Role-Based Security

While encryption ensures that data that



Site flow requiring encryption



New Web application using role-based security

is being transmitted over the network is neither being read nor tampered with by intermediaries, it does this universally for all users. Role-based security, on the other hand, is used to break users up into classifications based on their intended permissions and then to grant access to Web pages based on these classifications.

Role-based security should be used in one of two circumstances. First, when you have parts of the site that you wish to restrict to a

subset of all users. In our example we will show how to restrict part of the site to the "Customer" role, but other roles are possible. For example, one can imagine adding more functionality, such as the ability to search orders, to the site to support customer support employees.

The second circumstance where role-based security should be used is when you need to know the identity of the user. The username might be needed for complex

tasks such as maintaining statistics about which pages a user visits but also for simple tasks such as having a welcome message that includes the user name.

The username can be obtained by calling

```
request.getRemoteUser()
```

This method only returns the username after the user has logged in. Prior to logging in, the user is anonymous and `getRemoteUser()` returns null. The simplest way to force a user to log in is to have them access a protected page. This forces the container to execute the login process, after which `getRemoteUser()` will return a valid username.

In our example, `shoppingcart.jsp` requires login so that we can use the customer's username to look up their shopping cart in the database. While our sample does not actually access a database, it does print the username out in the header.

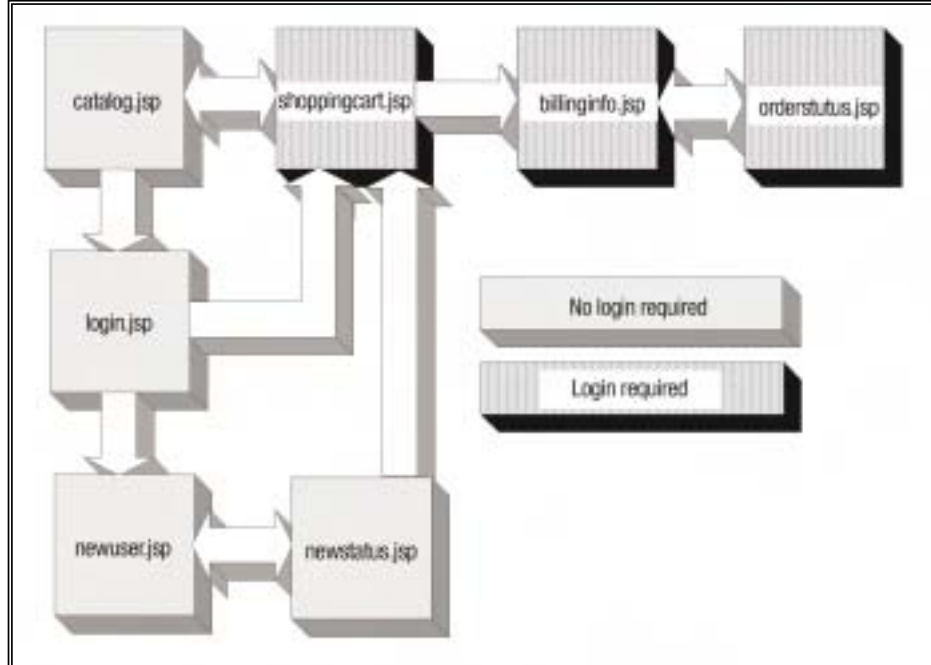
To add role-based security to the above sample, we will add the ability to maintain a permanent record of all customers' shopping carts and purchases. To accomplish this, the Web application must ensure that only authenticated users are allowed to access the shopping cart. Once the users are authenticated, the shopping cart and order information can be stored in a database associated with the user. Figure 2 illustrates the new Web application and the use of role-based security in it. Listing 2 is a fragment of the `web.xml` used to configure this security. One feature to note is that there is no need to add an explicit link to the `login.jsp` page from `catalog.jsp`. Simply link to `shoppingcart.jsp` and, due to the `<security-constraint>` and `<login-config>` tags, the servlet container presents the `login.jsp` page for you. Precisely, it will interpose the `login.jsp` form whenever `shoppingcart.jsp` is referenced (or any protected page for that matter) and the current user has not logged in. Furthermore, if the current user turns out not to be a Customer, then they will receive an access-denied error page (i.e., HTTP error code 403).

Declarative vs Programmatic Security

Specifying security in the deployment descriptor in this manner is called "declarative security." This is compared to "programmatic security," where the security is embedded in your code. Declarative security has some major advantages over programmatic security:

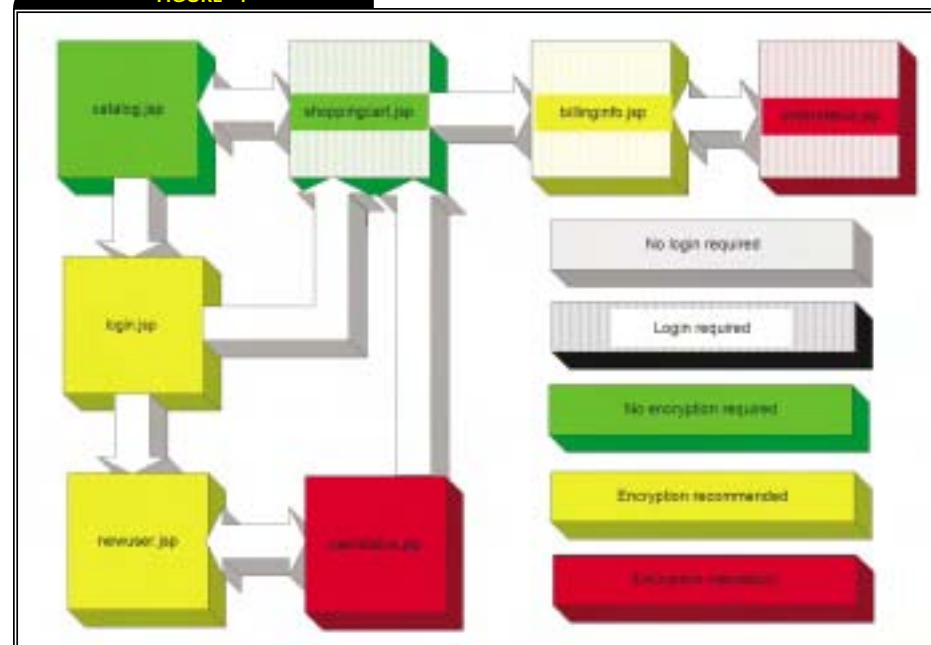
- **Simplicity:** Security code tends to be difficult to write and error prone. Declarative security allows you to avoid writing

FIGURE 3



New user registration

FIGURE 4



Completed application's flow

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS



3-Pack
Pick any 3 of our magazines and save up to **\$275⁰⁰**
Pay only \$175 for a 1 year subscription plus a **FREE CD**

- 2 Year – \$299.00
- Canada/Mexico – \$245.00
- International – \$315.00

6-Pack
Pick any 6 of our magazines and save up to **\$350⁰⁰**
Pay only \$395 for a 1 year subscription plus **2 FREE CDs**

- 2 Year – \$669.00
- Canada/Mexico – \$555.00
- International – \$710.00

9-Pack
Pick 9 of our magazines and save up to **\$400⁰⁰**
Pay only \$495 for a 1 year subscription plus **3 FREE CDs**

- 2 Year – \$839.00
- Canada/Mexico – \$695.00
- International – \$890.00

CALL TODAY! 888-303-5282

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.

TO ORDER • Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

MX Developer's Journal		
U.S. - Two Years (24) Cover: \$143	You Pay: \$49.99 /	Save: \$167 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$29.99 /	Save: \$60
Can/Mex - Two Years (24) \$168	You Pay: \$79.99 /	Save: \$137 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 /	Save: \$40
Intl - Two Years (24) \$216	You Pay: \$89.99 /	Save: \$127 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$59.99 /	Save: \$30
Digital Edition - One Year (12)	You Pay: \$19.99	

Linux World Magazine		
U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 /	Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

Java Developer's Journal		
U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

Web Services Journal		
U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

.NET Developer's Journal		
U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

XML-Journal		
U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

WebLogic Developer's Journal		
U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

ColdFusion Developer's Journal		
U.S. - Two Years (24) Cover: \$216	You Pay: \$129 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

Wireless Business & Technology		
U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$192	You Pay: \$139 /	Save: \$53 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$79.99 /	Save: \$16
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

WebSphere Developer's Journal		
U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

PowerBuilder Developer's Journal		
U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

*WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm



security code and simply “declare” your security requirements, the servlet container takes care of the rest.

- **Ease of maintenance:** When security is embedded in your program (i.e., programmatic security), a change in security means that code must be rewritten. With declarative security, frequently only the deployment descriptor needs to be altered. In our example, for instance, if the business requirements change such that only registered customers can access the catalog, the only change required would be to modify the `<security-constraint>` tag in the deployment descriptor.
- **Separation of responsibility:** Programming is typically done by engineers. Security is typically done by engineers, business analysts, application deployers, and others. If the application's security is embedded in the code, then only engineers can alter it. By embedding the security in the deployment descriptor, the security can be modified post-development without an engineer's involvement.

Online User Registration

Taking this example further, we can add pages to allow a new user to register. This flow is described in Figure 3. There is a “New User” link on the login.jsp form that takes the user to the newuser.jsp page. On this page, the user enters a username and password and then submits the form to the userstatus.jsp. If the user creation is unsuccessful, the userstatus.jsp will redi-

rect back to the newuser.jsp page, displaying an appropriate error message at the top of the page. If the user creation is successful, userstatus.jsp logs the user in, prints a success message, and provides a link for the user to continue to their original destination.

One thing to keep in mind as we extend this sample further is that while Figure 3 shows the login.jsp (and hence the newuser.jsp) page only being executed between catalog.jsp and shoppingcart.jsp, it is possible that it will appear in other places. Any time an unauthenticated user accesses a secure page, the login.jsp page can appear. Even if there are no links to the other pages (e.g., billinginfo.jsp), a user can type the URLs into a browser or they can bookmark the pages.

After starting out with some argument processing, userstatus.jsp then creates a user. J2EE provides no standard mechanism for user creation but BEA WebLogic Server does via the UserEditorMBean interface. Once a UserEditorMBean is acquired, all that is needed is to call

```
userEditorMBean.createUser(username, password,
    "");
```

(The third argument is descriptive text about the user and not used in this example.) This operation is protected and is available only to users in the Admin role. To allow userstatus.jsp to execute correctly, it needs to have a run-as tag to cause it to be run as an Admin user. The deployment descriptor code is:

```
<url-pattern>/billinginfo.jsp</url-pattern>
<url-pattern>/orderstatus.jsp</url-pattern>
<url-pattern>/shoppingcart.jsp</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>Customer</role-name>
</auth-constraint>
</security-constraint>

<login-config>
<auth-method>FORM</auth-method>
<form-login-config>
<form-login-page>/login.jsp</form-login-page>
<form-error-page>/login.jsp?errorMsg=Login error. Please try again.</form-error-page>
</form-login-config>
</login-config>

<security-role>
<role-name>Customer</role-name>
</security-role>
</web-app>
```

Listing 1

```
<web-app>
<security-constraint>
<web-resource-collection>
<web-resource-name>
Order Pages
</web-resource-name>
<url-pattern>/billinginfo.jsp</url-pattern>
</web-resource-collection>
<web-resource-collection>
<web-resource-name>
Order Pages
</web-resource-name>
<url-pattern>/orderstatus.jsp</url-pattern>
</web-resource-collection>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<user-data-constraint>
<transport-guarantee>
CONFIDENTIAL
</transport-guarantee>
</user-data-constraint>
</security-constraint>
</web-app>
```

Listing 2

```
<web-app>
<security-constraint>
<web-resource-collection>
<web-resource-name>
Customer Pages
</web-resource-name>
```

```
<servlet>
<servlet-name> userstatus </servlet-name>
<jsp-file> /userstatus.jsp </jsp-file>
<run-as>
<role-name> weblogic </role-name>
</run-as>
</servlet>
```

Remember to be sure to have created the “weblogic” user and put it in the “Administrators” group as described earlier to ensure this works.

Once the user is created, the servlet logs the new user in. Programmatically logging in a user is not supported by J2EE either, but WebLogic Server allows this via the following lines:

```
SimpleCallbackHandler callbackHandler =
    new SimpleCallbackHandler(username,
        password);
ServletAuthentication
    .authenticate(callbackHandler,
request);
```

Finally, userstatus.jsp displays a success message and a link to continue. The difficulty here is that the destination of the link could be any protected page and is not limited to shoppingcart.jsp. This is because the authentication process can begin any time a protected page is accessed. While J2EE provides no mechanism for discovering the destination URL, WebLogic Server gives access to this via the following call:

```
ServletAuthentication.getTargetURLForFormAuthentication(request.getSession())
```

One thing to note here is that to make this work, the new user must be put into the Customer role. This is accomplished in a two-step process. First, the Customer role is mapped to the Customers group (a group is a collection of users – be sure this has been created as described above). This mapping is done in the weblogic.xml file with the following lines:

```
<security-role-assignment>
<role-name>Customer</role-name>
<principal-name>Customers</principal-name>
</security-role-assignment>
```

Next, the Web application utilizes the WebLogic extension to put the new user into the Customers group in userstatus.jsp immediately after it is created by calling:

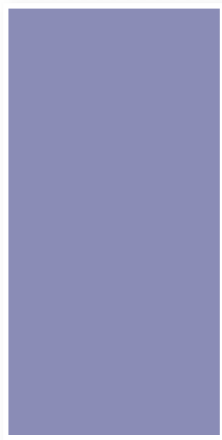
```
groupEditorMBean.addMemberToGroup("Customers",
username);
```

– continued on page 48

#1 Circulation in the World!

*JDJ is the highest circulation
i-Technology magazine in the world!*

162,019*



Java
Developer's
Journal

120,031*



Dr. Dobb's

75,561*



MSDN

*JUNE 2003 BPA AUDIT STATEMENTS *All circulation numbers are publishers' most current own data, six-month average circulation through June 2003.



Carmen Gonzalez
Senior VP Marketing



Miles Silverman
VP Marketing

Contact | Carmen Gonzalez: carmen@sys-con.com or 201 802.3021

JAVA DEVELOPER'S
JOURNAL



MANAGEMENT

The need for a server-side JVM is evident. The increase in the number of Java applications on the servers, and the exponential rise in the number of clients accessing these Java applications, brings forth the shortcomings in the traditional Java VMs, which are more tuned towards client-side processing.

code. Instead it uses a fast JIT compiler to compile all methods it encounters. The result of this is slower start-up times and a larger memory footprint, but faster execution times in the long run. During execution, a background thread is constantly looking for frequently run code. Such code is aggressively optimized and replaced during runtime. This is called adaptive optimization.

Runtime Management Console

JRockit comes with the JRockit JVM Management Console, which currently is not included with any of the commonly used JVMs. The management console comprises a management server and a stand-alone Swing application that is the console. You can turn on the management server by passing the following flag to the JVM on start-up:

```
"-XManagement"
```

An interesting note is that the console can be used with any JVM, not just JRockit. You can start the console by using the following command:

```
"java -jar <jrockit-install-  
directory>/console/ManagementConsole.jar"
```

Figure 1 shows how you can use the console to connect to the server.

I have changed the default Medical Records application that ships with BEA Platform 8.1 to use the JRockit JVM. Figure 2 shows the command window when the BEA WebLogic Server is starting up. Note on the top that I have enabled the JRockit management server. Towards the bottom of the window you can see that I am connecting to the management server using the management console.

When you talk about monitoring, there is always a performance impact that you have to accept when you are monitoring information. The JRockit management console can be run on the server or a remote machine, thereby reducing the overhead on the server. However, there is still some overhead on the monitored JVM. I recommend running the management console to monitor the JVM when needed (when you see problems) and then disconnect, so that there is minimal impact on the monitored JVM. The console has a notification feature that can trigger alerts based on thresholds for various variables. A few interesting events are

- CPU Load
- % of Used Heap
- Free Physical Memory

Management and Monitoring Using the JRockit JVM

ONE KEY TO BETTER PROBLEM DIAGNOSIS

BY KUNAL MITTAL

The first question that comes to mind when talking about JRockit is the comparison between the JRockit JVM and the Sun JVM. Although this article does not focus on this distinction, I will present a brief history of JRockit and then discuss the management and monitoring features introduced in it.

A few of the major issues with these JVMs are their ability to handle threads, the time taken for garbage collection, and the management of I/O operations. In a typical client-side application profile, 75% of the time a client JVM is running code, 20% of the time it's managing memory (garbage collection), and the remaining 5% of the time it's dealing with threads and I/O activity. On the server side the JVM spends about 25% of its time running code, 15% of its time managing memory, 30% managing threads, and 30% performing I/O.

JRockit was introduced by Appeal Virtual Machines, based in Stockholm, Sweden, and acquired by BEA in 2002. JRockit offers alternative thread management and memory management to fit the needs of these server-side applications. It also introduces a more adaptive code optimization based on runtime monitoring information.

JRockit is fully J2SE certified. A handful of core classes have been modified for better performance, but the majority of classes are unchanged from the Sun JVM.

Interpreter Versus Compiler

A key distinction between JRockit and traditional JVMs is that JRockit does not interpret

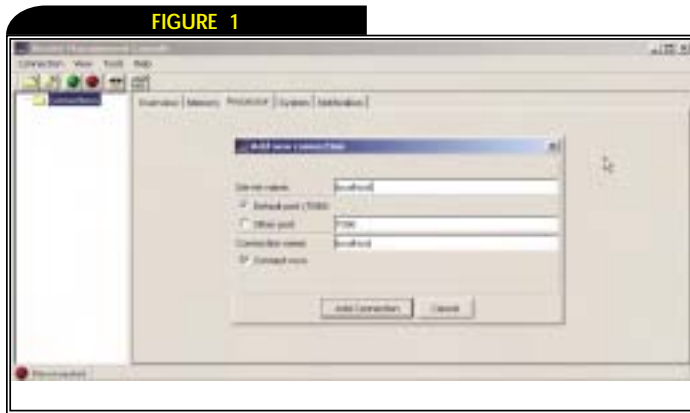


AUTHOR BIO

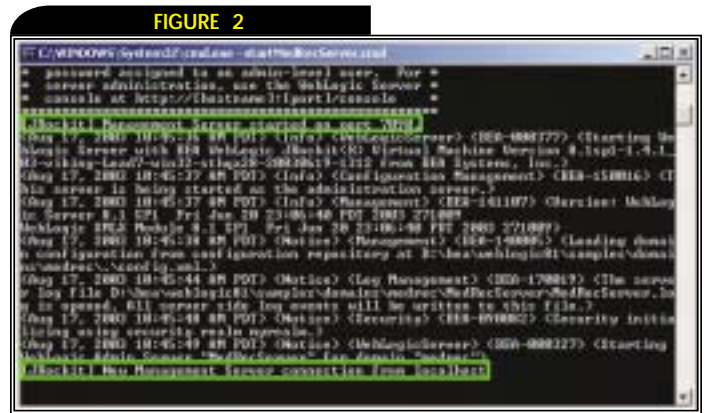
Kunal Mittal is a solutions engineer at Wakesoft, Inc., and a consultant for implementation and strategy for Web services and services-oriented architectures. He has coauthored and contributed to several books on J2EE, WebLogic, and Web services. Over the past several years, Kunal has worked on numerous projects using different BEA products.

CONTACT...

kunal@kunalmittal.com



JRockit connection



WebLogic startup command window

- JVM Absolute CPU Load
- % of Used Memory
- Average Time Spent in GC

Figure 3 shows some of the monitoring information that you can see regarding the Medical Record application as users are connecting to it.

The console also allows profiling of user-specified methods, showing the number of times a given method is invoked. This is roughly like a profiler, but the user must specify the class and method. Similarly, the console can be used to count the number

of times an exception was thrown. The user must specify the exception to be counted.

Integration with the WebLogic Console

Many developers, and especially the operations departments, complain that they have to use still another console. This is not true with JRockit. The standard WebLogic console integrates with the JRockit management server. You can get the same monitoring capability that you get using the JRockit management console through the standard BEA WebLogic con-

sole. However, remember that the WebLogic console can also be used to monitor the JRockit JVM, not any other JVM. This is primarily due to the lack of available monitoring features in other JVMs.

JRockit Runtime Analyzer Tool

The JRockit JVM is instrumented to provide runtime behavior regarding the Java application. Such information was not available previously. Various tools take an outside-in, after-the-fact approach on trying to extract such information by hooking into the JVM. The inside-out approach taken by JRockit provides more accurate information with less performance impact.

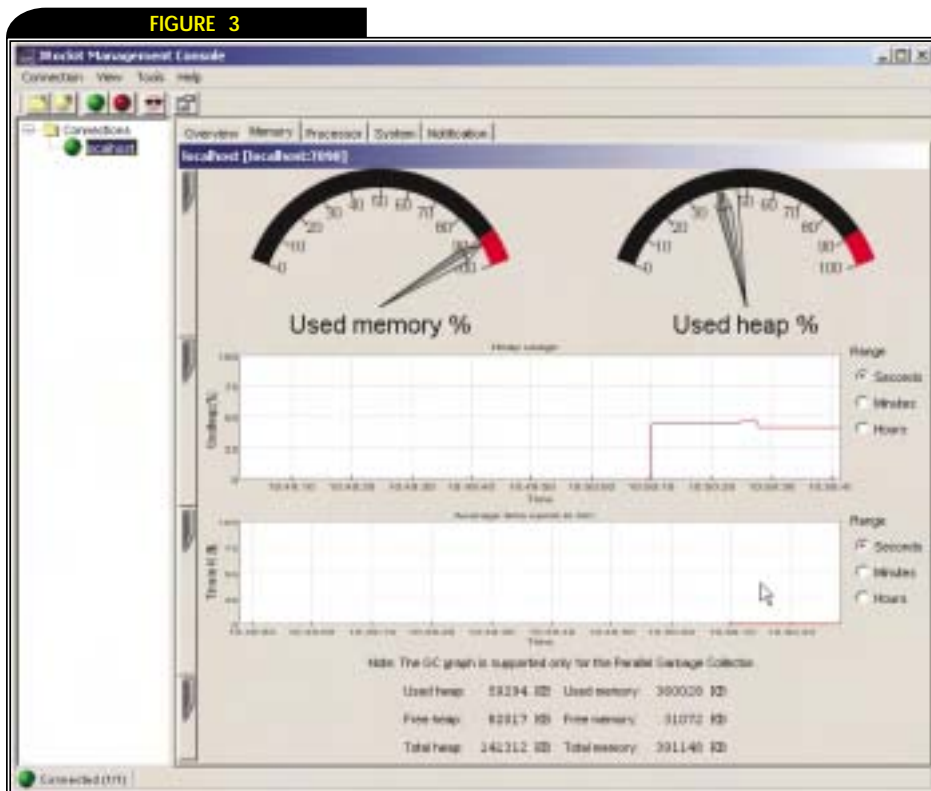
The JRockit Runtime Analyzer is used by BEA engineers internally to tune and refine the JVM. Only recently has BEA opened this to the entire BEA community. You can download this tool from the BEA site and use it to analyze your own Java applications.

Summary

This article talked about the management and monitoring capabilities of the JRockit JVM. Although monitoring always has some impact on performance (and don't believe anyone who says otherwise), the ability to monitor your applications through your development life cycle and in production environments is key. Monitoring allows you to quickly identify and pinpoint problems and defects. The rich data that monitoring can provide can have a huge impact on the overall robustness of your applications.

References

- *BEA WebLogic JRockit 8.1 on Dev2Dev from BEA:* <http://dev2dev.bea.com/products/wjrockit81/index.jsp>



Monitoring information

Cyanea/One from Cyanea

AN OUT-OF-THE-BOX TOTAL PRODUCTION MONITORING SOLUTION

Reviewed by Jason Snyder



Suppose you've developed your suite of applications, standardized to J2EE, and are now awaiting the J2EE benefits for monitoring these applications. You have a consistent series of applications, so adding advanced monitoring capability should be fairly straightforward. Unfortunately, you soon find that frequently this is not the case. Most monitoring applications are built to support a variety of architectures. This may be a strength for some production environments, but as a consequence, these solutions aren't taking advantage of much of what J2EE offers.

Cyanea is now a BEA Star Partner and Cyanea/One provides consistent capability for both application servers. Once installed, Cyanea/One provides a suite of features that offer automatic coverage for all of your J2EE applications – no intrusive instrumentation is needed. It also works independently of your J2EE applications so there is no need to modify or even access your application code. And even though you have done nothing to your applications, you have the ability to view them from an enterprise level all the way down to specific method or thread-level detail.

I recently had the opportunity to try out Cyanea/One on the BEA WebLogic Server and was fairly impressed. You get a sense of the overall high quality of the product when the initial application overview screen appears. It is a very clean, easy-to-read view of all application server groups and their associated volume throughput (see Figure 1). If there is a problem with one of the applications, simply drill down on that application and additional detail is provided. The application server groups can be customized and removed from the view as needed. In addition to volume throughput, you can see the average response time for the group as well as a comparison of that response time to historical performance. It would be nice if additional statistics were available as alternative high-level application views, but the view gives you enough to know that something could be wrong or that everything is working fine.

I created a problem: a vast number of requests was assaulting the database for the sample application! Sure enough, the application overview showed that there was a problem and I drilled down to determine the problem. The next view, "In-

Flight Request Search," lists all of the current requests being processed by the application server. You can easily sort it by the total resident time to determine any long-running transactions. Hung transactions or threads are easily called out within this view as well.

You can then drill down on the particular thread that interests you and the resulting "Request Detail" provides a summary of the thread, including the last SQL that was run. From there, specific methods can be viewed, and requests can be cancelled or reprioritized. The drill-down capabilities are intuitive and it is fairly easy to determine the cause of most of the problems. The statistics for these views are not configurable, but seem to provide most values that you would need to do standard analysis. For example, at the method level Cyanea shows you the entry/exit points along with corresponding response times and CPU times for that method invocation, in addition to the SQL statements called within that method.

Another nice view of an application server or server group is the System Resources overview (see Figure 2). This view provides an easy-to-understand graphical view of the major components of the application server and allows a quick understanding of possible problem causes. JVM CPU usage, JVM memory usage, database connection pools, and transaction failure values are provided, and each can easily be drilled down for additional detail.

I was really impressed with the variety and usefulness of the standard reports. The reports are built within the overall application paradigm, so drilling down is a standard capability (see Figure 3). Custom reports based on spe-

Licensing Info:

Cyanea/One is licensed by number of CPU's per managed server running in the customer environment. [so for example a 4 CPU server would require 4 licenses; the managing device or "Managing Server" is included at no charge, regardless of the number of CPU's]

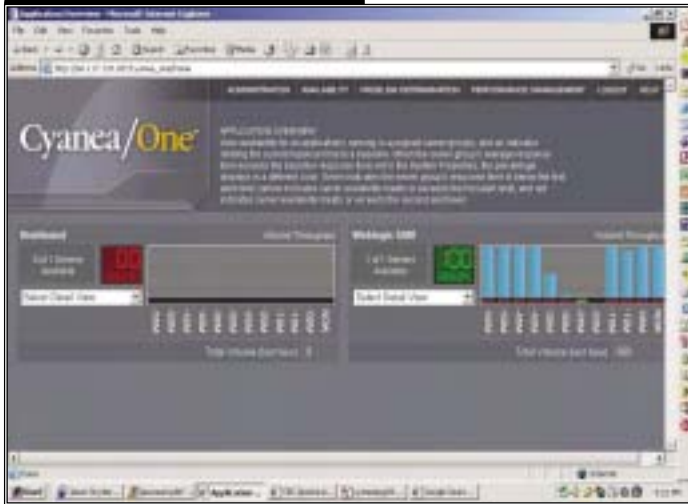
System Requirements:

Managing Server:
OS: Linux, Sun Solaris, IBM AIX
Database: Oracle, DB2

Managed Servers:

OS: Linux, MS Win2K Server, Sun Solaris, HP-UX, IBM AIX, z/OS, and OS/390
Database: Oracle, DB2

FIGURE 1



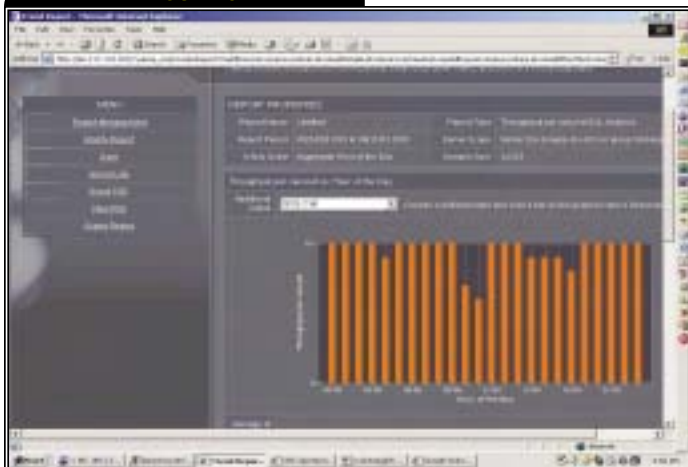
Application overview screen

FIGURE 2



System Resource overview

FIGURE 3



Drilling down into the application

cific metrics can easily be created for possible uses like memory leak analysis, or capacity planning and workload characterization efforts. Another reporting feature I liked was the ability to e-mail any report in PDF format. As anyone who has been on late night call can attest, nothing beats the ability to see the data instead of having support describe the situation.

The same reporting area, "Performance Analysis and Reporting," allows access to historical information. Historical transactional data is stored at the detail level using unique instances of each request, not an aggregate thereof. The user can define what is stored. All requests against all servers can be stored, but it is generally recommended that only a certain percentage be saved. Each customer can specify the specific percentage based on server status. In the current version, this percentage is a global setting; the next release (2.1) provides a percentage setting for each server and is shipping now. In addition, server availability, transaction throughput, and response times are continuously monitored against historical performance and can be graphically displayed.

A related feature is called "Monitoring on Demand" and provides a variable server monitoring level to minimize impact. When everything is running smoothly, less information needs to be persisted than when there are known problems.


Another useful area is the Server Statistics overview that allows you to display customizable metrics for particular servers of interest, as well as select thresholds that can visually alert you when they are exceeded. Other thresholds can be set with the Traps and Alerts area that can notify you if a customizable system or application condition is met. You can trap

not only for "standard" events such as heap size utilization or DB connection pool percentage, but also for application-specific events such as "request A is hung for more than x seconds" or "method y is invoked more than z times in a single request."

It is important to note that Cyanea does not support stand-alone Java applications (non-J2EE) at this writing. The next version will display non-J2EE threads that are started by J2EE applications, but the reality is that Cyanea is a J2EE application monitoring solution.

Conclusion

For a J2EE production environment, Cyanea/One offers an out-of-the-box total production monitoring solution. Installation and setup are straightforward and easy. Any J2EE application within your enterprise suite can be included automatically once Cyanea/One is installed.

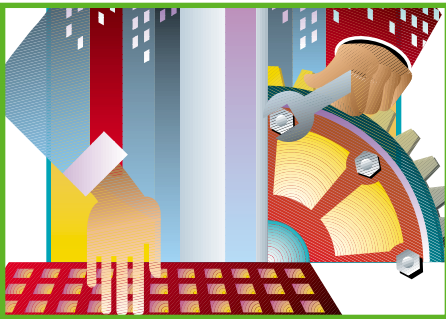
Cyanea/One also provides a thin-client interface, advanced memory/heap analysis features, granular historical data collection, and the ability to change the monitoring scope at a JVM level dynamically, and to do so without having to restart the app server. It truly is a complete J2EE application monitoring solution. 

AUTHOR BIO

Jason Snyder is the product review editor for *WebLogic Developer's Journal*. He is an architectural expert for CSC Consulting in Boston, and has served as the lead architect for several J2EE development projects. He has over 10 years of experience in software development, OO design, and application architecture.

CONTACT...

jasonsnyder@townisp.com



PERFORMANCE

How to Diagnose a Performance Problem in a J2EE System

THE CURE MAY BE SIMPLER THAN YOU THINK

BY JOHN BLEY



AUTHOR BIO

John Bley is a software engineer for Wily Technology. He has extensive experience with Java programming and architecture. For this article, he has drawn on the experiences of Wily's enterprise customers who are responsible for managing complex J2EE environments.

CONTACT...

jbley@wilytech.com

So you've been told to diagnose a performance problem in a WebLogic J2EE application. Because Java systems are so complex, this can be a bit like diagnosing a rare illness.

To pinpoint the problem accurately you need to have a thorough understanding of the symptoms, be prepared to do a fair amount of investigative work, and then you must determine the proper remedy. This article offers a discussion of some of the most common types of J2EE application performance issues and their causes, followed by suggested guidelines for properly diagnosing and eliminating them.

The Symptoms

What are the symptoms of a WebLogic application performance problem? The symptoms you see guide your search through all possible illnesses. Get a notebook and start asking people for data. Try to separate speculation and theory about the root cause of the problem from hard evidence about the system's behavior. Here's a list of common symptom sets:

- **Consistent slowness:** The application is simply always too slow. Changing environmental factors (load, number of database connections) doesn't change the overall response time much.
- **Slower and slower over time:** The system gets slower and slower the longer it runs (under a fairly consistent load). Perhaps a threshold is (eventually) reached and the system locks up or melts down with a deluge of errors.
- **Slower and slower under load:** The applica-

tion keeps getting slower and slower with each additional user. If users are kept off the system, it "cools down" and returns to normal.

- **Sporadic hangs or aberrant errors:** Occasionally (perhaps based on load or some other condition), users see hangs where pages simply never complete or error pages with an exception and a stack trace. The number of hangs may vary a bit, but never seems to go away completely, even after a "burn in" period.
- **Foreseeable lock ups:** Hangs or errors happen a bit at first, but accelerate over time until the system locks up entirely. Typically these are accommodated with "management by restarts."
- **Sudden chaos:** The system runs fine, with more or less acceptable and consistent performance for some period of time (could be an hour, could be three days), when "suddenly, for no reason at all," it starts spewing errors, or else locks up.

Why Is Problem Diagnosis So Complicated?

There is no set formula that you can apply to derive the performance for a particular usage pattern of a WebLogic application (in hard real-time engineering, techniques like rate monotonic analysis can do exactly this, but let's ignore that for the purposes of this article). The resulting performance is sensitive to whether or not another system somewhere else on the network is hitting a shared back-end service hard. Perhaps it's also dependent on matching the exact versions of the JDBC driver and the database. Maybe a developer wrote some code three years ago that happened to swallow a particular type of exception and the feedback you desperately need to solve the problem is contained in that exception.

In essence, the performance of a typical business system is an emergent property resulting from thousands of interacting variables and decisions. Like a human body, there are too many interlocking parts and processes to comprehend the totality of the domain. So we simplify, and look for overarching patterns.

The Diseases

What are the possible root causes for the symptoms you're seeing? Is it your basic flu or the beginnings of pneumonia? Is the underlying problem internal to the application or is it external to its JVM? See Table 1 for some of the most common causes of poor application performance.



Measuring Vital Statistics

As the person charged with diagnosing the problem, you should be able to keep track of vital statistics about the health of your WebLogic application. What can you measure? What tools are available to help?

- **Total memory in use:** At various levels (JVM heap, OS), Java heap profilers provide visibility into the exact usage of the heap; tools like top, vmstat, and Windows Perfmon give visibility into memory usage at the OS level. For a simple aggregate view of the Java heap try turning on `-verbose:gc` if available in your JVM.
- **CPU time:** In aggregate (available via top and so forth), per component or per method. Some of these metrics can be obtained through the WebLogic Admin Console. They are also available via a Java profiler.
- **Wall-clock time (a.k.a. "real" time):** Per transaction, per component, per method; viewable as statistical averages or individual data points. Java profilers can yield some of this data, though an application monitoring solution is probably your best bet.
- **Internal resources:** Number allocated, number in use, number of waiting clients, average wait time to obtain a resource, average time spent using the resource, average time it takes the resource to accomplish requested work. Application servers typically give some minimal visibility into these numbers.
- **External resources:** Number allocated, number in use, number of waiting clients, average wait time, plus measurements directly on the external system such as its view of how quickly it's completing requested work. Don't forget that the operating system and hardware that the application server run on are "external resources" as well – e.g., are you using too many processes or ports? Measuring these resources comes in two forms – measuring the bridge layer to that resource from inside the JVM and measuring the external resource with a tool native to that resource.
- **Network utilization:** Bandwidth usage, latency. Network sniffers and equipment give insight into this, though OS-local tools like netstat can help too.
- **System state:** Use thread dumps, logs and trace files, stack traces, etc. Or at a deeper level, use values of variables as seen under a debugger.

TABLE 1

"DISEASE"	DESCRIPTION	SYMPTOMS	CAUSES OR CURES
Linear memory leak	A per-unit (per-transaction, per-user, etc.) leak causes memory to grow linearly with time or load. This degrades system performance over time or under load. Recovery is only possible with a restart.	Slower and slower over time Slower and slower under load	This is most typically linked with a resource leak, though many exotic strains exist (for example, linked-list storage of per-unit data or a recycling/growing buffer that doesn't recycle).
Exponential memory leak	A leak with a doubling growth strategy causes an exponential curve in the system's memory consumption over time.	Slower and slower over time Slower and slower under load	This is typically caused by adding elements to a collection (Vector, HashMap) that are never removed.
Bad coding: infinite loop	Threads become stuck in while(true) statements and the like. This comes in CPU-bound and wait-bound/spin-wait variants.	Foreseeable lockup	You'll need to perform an invasive loop-ectomy.
Resource Leak	JDBC statements, CICS transaction gateway connections, and the like are leaked, causing pain for both the Java bridge layer and the back-end system.	Slower and slower over time Foreseeable lockup Sudden chaos	Typically, this is caused by a missing finally block or a more simple failure to close() the objects that represent external resources.
External bottleneck	A back end or other external system (e.g., authentication) slows down, slowing the J2EE app server and its applications as well.	Consistent slowness Slower and slower under load	Consult a specialist (responsible third party or system administrator) for treatment of said external bottleneck.
Over-usage of external system	The J2EE application abuses a back-end system with requests that are too large or too numerous.	Consistent slowness Slower and slower under load	Eliminate redundant work requests, batch similar work requests, break up large requests into several smaller ones, tune work requests or back-end system (e.g., indexes for common query keys), etc.
Bad coding: CPU-bound component	This is the common cold of the J2EE world. One bit of bad code or a bad interaction between bits of code hogs the CPU and slows throughput to a crawl.	Consistent slowness Slower and slower under load	The typical big win is a cache of data or of performed calculations.
Layer-itis	A poorly implemented bridge layer (JDBC driver, CORBA link to legacy system) slows all traffic through it to a crawl with constant marshalling and unmarshalling of data and requests. The disease is easily confused with External Bottleneck in its early stages.	Consistent slowness Slower and slower under load	Check version compatibility of bridge layer and external system. Evaluate different bridge vendors if available. Re-architecture may be necessary to bypass the layer altogether.
Internal resource bottleneck: over-usage or under-allocation	Internal resources (threads, pooled objects) become scarce. Is over-utilization occurring in a healthy manner under load or is it because of a leak?	Slower and slower under load Sporadic hangs or aberrant errors	Under-allocation: increase the maximum pool size based on highest expected load. Over-usage: see Over-usage of External System.
The Unending Retry	This involves continual (or continuous in extreme cases) retries of a failed request	Foreseeable lockup Sudden chaos	It might just be that a back-end system is completely down. Availability monitoring can help there, or simply differentiating attempts from successes.
Threading: chokepoint	Threads back up on an over-ambitious synchronization point, creating a traffic jam.	Slower and slower under load Sporadic hangs or aberrant errors Foreseeable lockup Sudden chaos	Perhaps the synchronization is unnecessary (with a simple redesign) or perhaps more exotic locking strategies (e.g., reader/writer locks) may help.
Threading: Deadlock / Livelock	Most commonly, it's your basic order-of-acquisition problem.	Sudden chaos	Treatment options include master lock, deterministic order-of-acquisition, and the banker's algorithm.
Network saturation	High latency or basic inability to route all the requested packets results in timeout exceptions, hangs, livelock, etc.	Consistent slowness Slower and slower under load Sporadic hangs or aberrant errors	This may be a troublesome system flooding the network, or it could be time to upgrade the infrastructure (either a faster network or a router with better throttling to avoid similar problems in the future)

Some common causes of poor application performance



Lab Work

Sometimes the data obtained during one benchmark run will not reveal the answer. And chances are that you will have a limited budget for running experiments and doing lab work to complete your diagnosis. What kinds of experiments can you run? What variables can you change or watch?

• **Try watching the system's behavior over time.** Apply a constant load and watch how your measurements vary as time passes. You might see some trends in as little as an hour, or they might take a couple of days. For instance, you might see memory usage increasing over time. As the amount of memory in use reaches its upper limits, the time the JVM spends garbage collecting and the

TABLE 2

Load (# users)	Round-trip wall-clock time (ms)
10	300
50	471
100	892
150	1067

Application performance is slower and slower under load.

TABLE 3

Load (# users)	Round-trip wall-clock time (ms)	Aggregate CPU time (%)
10	300	30
50	471	33
100	892	37
150	1067	41

The problem seems to be a "waiting" bottleneck

TABLE 4

Load (# users)	# of threads waiting for database connection	JDBC query wall-clock time (ms)
10	2	58
50	3	115
100	3	489
150	4	612

Is the problem a slow SQL statement?

TABLE 5

Load (# users)	JDBC query wall-clock time (ms)	DB SQL execution wall-clock time (ms)
10	58	43
50	115	97
100	489	418
150	612	589

The SQL statement is actually slow in the database.

time the OS spends thrashing the memory pages around starts to weigh down the aggregate response time of your users' transactions. Full garbage collections might be so long as to cause timeouts and exceptions on any in-flight transactions when the GC was kicked off. Start looking for a resource or memory leak.

• **Try varying the load on the system.** Pick three or four workloads (say, 10, 50, and 100 users) and collect data at each of them. Plot your measurements against these loads. You may find, for example, that while your account login servlet responds in less than 50 milliseconds no matter what, the sales tax calculation EJB gets slower and slower linearly with an increasing number of users. If the difference in that EJB's performance under load explains the bulk of the aggregate response time's performance under load, then it's time to dig deeper into that component. Be sure to also try backing off the load and seeing if the system recovers or not.

• **Try compartmentalizing the system and stressing the individual parts in turn.** Pick one or more axes of compartmentalization. The primary one is tiers of the system: load balancer, Web server, WebLogic Server, and back end. Other examples include user accounts, internal components, transaction types, and individual pages. Let's say you pick user accounts. Run some load under user A's account and then some load under user B's account (which is hopefully quite different); compare various measurements between the two runs. Or, take the back-end systems you use in turn and stress parts of the application that use each back end heavily. Which axis of compartmentalization you pick will depend entirely on which theory you're trying to prove or disprove. Here are some ideas:

- If a particular user's logon seems to trigger a problem, it might be that user's account profile (e.g., loading the full purchase history of 2,000 orders), or it might be the way he uses the system (e.g., order of page accesses or exact query string he uses to search for a particular document).
- If you've got a clustered system, try compartmentalizing by individual machines. Despite best efforts, sometimes boxes don't have the latest app server or OS patches, which can contribute to different performance characteristics. Also, pay attention to the load balancer or nanny process to see if it's distributing

work fairly and keeping up with inbound requests.

Diagnosis: Testing Your Theories

At this point, you should have enough information to form theories about the cause of the performance bottleneck (see Table 1). To confirm that your theory is correct or to differentiate between multiple competing theories, you'll need to analyze more information or run additional benchmarks on the system. Here are a few guidelines to help you out.

• **To distinguish between poor coding (either an application component or layer-itis) and a bottleneck (internal or external),** try looking at aggregate CPU usage. If it doesn't vary under load but overall response time does, then the application is spending most of its time waiting.

• **Just because it appears to be a problem with an external resource doesn't mean you can immediately blame it on that resource.** Layer-itis or a networking problem, for example, can cause the database to appear slow even though it's not. Or, more simply, your demands on that database could be unreasonable (2 million row joins across three tables, each time a user logs in). Always compare response times at the bridge layer (e.g., JDBC driver) with those provided by the resource or a tool native to it (e.g., DBA Studio).

• **Architectural diagrams help you understand overall interactions inside the system, but don't forget that the map is not the territory.** Coding mistakes or misunderstandings of architectural intent may make the actual behavior of the system vary from what's expected. Trust hard numbers from a performance tool more than a document claiming that only one SQL statement will be issued per user transaction.

• **Apply Occam's razor:** Let's say your two competing theories are that either there's a poorly coded component in the 2 million lines of code system that wasn't integrated until last week, or that the JVM's Just-In-Time compiler is creating bad machine code which is destroying the memory integrity of this variable. Unless you have specific data to prove otherwise (and I have seen the second happen), investigate the first theory more fully than the second one. J2EE systems are prone to byzantine failures, but don't let that dissuade you from testing a simpler theory first.

SYS-CON MEDIA

304,187 of the World's Foremost IT Professionals

DIRECT MAIL, EMAIL OR MULTI-CHANNEL

Target CTOs, CIOs and CXO-level IT professionals and developers who subscribe to SYS-CON Media's industry leading publications

Java Developer's Journal...
The leading publication aimed specifically at corporate and independent java development professionals



LinuxWorld Magazine...The premier monthly resource of Linux news for executives with key buying influences



Web Services Journal...The only Web Services magazine for CIOs, tech, marketing & product managers, VARs/ISVs, enterprise/app architects & developers



XML-Journal...The world's #1 leading XML resource for CEOs, CTOs, technology solution architects, product managers, programmers and developers



PowerBuilder Developer's Journal...The only PowerBuilder resource for corporate and independent enterprise client/server and web developers



WebSphere Developer's Journal...The premier publication for those who design, build, customize, deploy, or administer IBM's WebSphere suite of Web Services software



ColdFusion Developer's Journal...The only publication dedicated to ColdFusion web development



WebLogic Developer's Journal...The official magazine for BEA WebLogic application server software developers, IT management & users



.NET Developer's Journal...The must read iTechnology publication for Windows developers & CXO management professionals



Wireless Business & Technology...The wireless magazine for key corporate & engineering managers, and other executives who purchase communications products/services

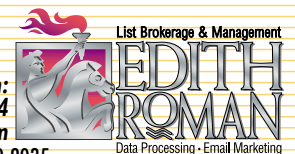
Recommended for a variety of offers including Java, Internet, enterprise computing, e-business applications, training, software, hardware, data back up and storage, business applications, subscriptions, financial services, high ticket gifts and much more.

NOW AVAILABLE!
The SYS-CON
Media Database
304,187 postal
addresses



For email information:
contact Frank at 845-731-3832
frank.cipolla@epostdirect.com
epostdirect.com 800-409-4443 fax 845-620-9035

For postal information:
contact Kevin at 845-731-2684
kevin.collopy@edithroman.com
edithroman.com 800-223-2194 fax 845-620-9035





• **Absence of errors in the log file does not imply absence of errors.** Exceptions don't get written to logs for many reasons; perhaps a programmer thought that something would "never happen" and swallowed the exception, or perhaps some component has a fallback mechanism it can use and therefore doesn't log the first failure.

Example Diagnosis

Let's step through an example. Your WebLogic application exhibits the symptom of increasing slowness under load. The more users you pile on, the slower things get. Once the load is removed, the system cools down with no side effects. You measure this primary symptom and find the following (time measurement is for end-to-end completion of a single typical transaction) and get the results shown in Table 2.

You form a few theories. Perhaps the disease here is a badly coded component or perhaps it's a bottleneck on a back-end system. It could be a synchronization choke-point. How can you tell the difference? Suppose you also measured aggregate CPU usage of the application server during the

load runs and got the results shown in Table 3.

It looks like the system isn't CPU bound, which means that it's spending most of its time waiting. But is it internal (a synchronization traffic jam, for instance) or external (slow database)? Well, let's suppose we gather a few more numbers and come up with Table 4.

It doesn't appear to be an internal bottleneck waiting for database connections – instead, it appears to be the JDBC query itself. Not only does the JDBC query vary with the overall transaction time, but its poor performance explains the bulk of the overall poor performance. We're still not quite done, though. You still have three major theories to sort through: Is it the database itself that's slow, is the application making unreasonable demands on the database, or does the problem lie in some layer between the application and the database? You pull up the database's vendor-specific tool to see response times from its point of view. You'd hope to see numbers such as those in Table 5.

If you didn't see this information, then you might dive back into the JDBC driver and hope to find some sort of synchronization problem inside it (remember, the CPU isn't overwhelmed). Fortunately, in this case you've narrowed the specific problem down to a database query. Figuring out if the query's demands are reasonable enough requires some domain knowledge and familiarity with the system, but perhaps in this case it simply turns out that the query compares an unindexed field against a foreign key. You work with the DBA, who changes the indexing scheme to make this query faster, and you've found your cure.

Conclusion

Diagnosing a performance bottleneck in a WebLogic J2EE application can be a difficult journey. Keep your wits about you, separate fact from speculation, and always confirm your hypotheses with hard evidence. Hopefully I've given you a taxonomy of useful ideas to think about and experiment with. Like debugging, this is still a black art but careful thinking will see you through. Good luck! 🍀

SECURITY

– continued from page 38

Programmatic Security

Programmatic security, while supported by J2EE, is risky business. It tends to lead to inflexible and error-prone code. One good use of the programmatic security APIs, though, is for personalization. Imagine extending our example above to include a link off the catalog page that allows you to update the catalog entry via `updateentry.jsp`. This link is applicable only to users in the `CatalogAdmin` role. The following code in `catalog.jsp` would introduce a conditional link for `updateentry.jsp`. This link will be visible only to users in the `CatalogAdmin` role in much the way the "Logout" link is visible only when a user is logged in.

```
<% if (request.isUserInRole("CatalogAdmin")) { %>
  <a href="updateentry.jsp?productid=<%=getProductID()%>">
    Update Catalog Entry</a>
<% } %>
```

This code assumes that `updateentry.jsp` takes a `productid` argument and that a `getProductID()` method exists to return the ID of the product currently being displayed.

Bringing It All Together

Figure 4 illustrates the completed application's flow describing both encryption and role-based security. While encryption and role-based security must be combined to ensure security in a

Web application, they are used for different purposes. Encryption is used to ensure security of data on the network. Role-based security is used to categorize users and grant permissions to them based on their categories. J2EE provides mechanisms of enabling both of these security features via the Web application's `web.xml` file. Furthermore, J2EE provides some standard mechanisms for programmatic security, such as accessing the user's name.

BEA webLogic Server's security extensions, such as the `UserEditorMBean` and the `GroupEditorMBean`, can be used for user management. User management features are not included in the J2EE standard. In our example, we used these user management features to implement online user registration, a feature common to many Web sites.

BEA WebLogic Server also extends programmatic security, enabling a programmatic mechanism of logging into the server. By using `ServletAuthentication.authenticate()` and `ServletAuthentication.getTargetURLForFormAuthentication()`, our sample demonstrated how a user could be programmatically logged in and redirected to their original destination.

These WebLogic Server extensions are designed to dovetail with the security features available for Web applications in J2EE. They fill gaps in the standard J2EE security functionality, allowing the implementation of richer and more functional Web applications. 🍀

THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

**SPECIAL
OFFER!**
SAVE \$31*
OFFER SUBJECT TO
CHANGE WITHOUT
NOTICE



Helping you enable
intercompany
collaboration on
a global scale

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

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.



WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Actional	www.actional.com/guide	650-254-4100	29
BEA Systems	http://dev2dev.bea.com/medulla	800-817-4BEA	2, 3
Confluent Software	http://www.confluentsoftware.com	408-523-9000	51
Cyanea	www.cyanea.com/wldj/nomoremadness.html	877-CYANEA8	11
Dirig Software	www.ebizperform.com	603-889-2777	9
Hewlett-Packard	www.hp.com/plus_fedex	800-752-0900	19
Java Developer's Journal	www.javadevelopersjournal.com	888-303-5282	39
Mercury Interactive	www.mercuryinteractive.com/optimizej2EE	800-831-8911	52
Quest Software	http://java.quest.com/jprobe/wldj	800-663-4723	13
Quest Software	http://java.quest.com/jcsv/wldj	800-663-4723	23
ReportingEngines	www.reportingengines.com	888-884-8664	17
SYS-CON Media Database	www.edithroman.com,	800-223-2194,	47
SYS-CON Media Database	www.epostdirect.com	800-409-4443	47
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	37
SYS-CON Reprints	www.sys-con.com	201-802-3026	33
Web Services Edge East 2004	www.sys-con.com	201-802-3069	31
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	49
Wily Technology	www.wilytech.com	888-GET-WILY	5, 25

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

Introduction To ebXML in WebLogic Integration 8.1

With today's increasing demand for businesses to communicate with each other, B2B Integration holds the key to successful e-commerce collaboration. ebXML-Messaging provides an XML-based infrastructure to enable a consistent, secure, and interoperable message exchange.

Performance Improvement of a J2EE Application

In the real world, managing a production system is all the more difficult when the future environment in which the system will run isn't taken into account in the design phase. System re-engineering is a simple task when consideration is given to the boundaries and context of the application.

The Cost of Marshalling

The cost of marshalling may not be as expensive as most of us thought it would be. Passing objects by reference when in the same .ear file is the most efficient way to handle it.

Create Command-line Scripts Out of Your Domain Configuration

conf2admin is a utility that uses XSLT to generate a set of command-line scripts for a WLS domain configuration. It enables a WLS administrator to generate a script to reproduce a WLS domain at a later date.

Configuring WebLogic 7.0 JDBC Connectivity

This tutorial explains the configuration of WebLogic 7.0 JDBC with the Oracle 8.1 database server. It will include sections on the connection pool, multipool, datasources, and datasource factory

WebLogic
DEVELOPER'S JOURNAL

News & Developments

BEA Launches Developer Extensibility Program

(San Jose, CA) – BEA Systems, Inc., the world's leading application infrastructure company, has launched the BEA Controls and Extensibility Program, which is designed to help independent software vendors (ISVs) easily and quickly integrate with BEA WebLogic Platform 8.1.

The new program provides ISVs with free software, technical resources, testing services, and marketing support to help make it easier for them to reach Java developers and accelerate the adoption of their applications within the BEA customer base. Developers will find it easier to customize the WebLogic Workshop development environment to better suit their needs and provide a standard infrastructure for building and integrating applications and technologies within the WebLogic Platform.



www.bea.com

Actuate Integrates e.Report Engine with WebLogic Workshop 8.1

(South San Francisco) – ReportingEngines, a division of Actuate Corporation and provider of embedded reporting solutions for the J2EE platform, has released the Formula One e.Report Engine for BEA WebLogic Workshop. The e.Report Engine for BEA WebLogic Workshop offers developers a fully integrated Java reporting toolset that can be used as part of every project to design, preview, compile, and deploy reports for any application without leaving the WebLogic Workshop environment.



Developers will utilize a fully integrated WebLogic Workshop

Control that enables reports to be built in a wizard-driven, point-and-click manner from JDBC, BEA Liquid Data for WebLogic, XML data streams, and in-memory Java objects.

www.ReportingEngines.com,
www.actuate.com

SandCherry Releases AppDev VXML

(New York) – SandCherry, Inc., a provider of software for cost-effectively developing, testing, and deploying advanced speech-enabled services, has announced the availability of AppDev VXML, an integrated voice-application development tool designed for BEA WebLogic Workshop 8.1. AppDev VXML reduces the time, effort, and complexity of building a speech application by reusing the same underlying business logic and data integration implemented for a Web application.

Unlike other solutions requiring entirely new business logic and data integration for a speech application, AppDev VXML simply adds a new voice-based presentation layer. Changes to business logic or data integration simultaneously update both Web and speech applications, reducing ongoing application maintenance costs.



www.sandcherry.com

U.S. Navy Upgrades NKO Portal to Appian Enterprise

(Vienna, VA) – The U.S. Navy has awarded Appian Corporation, a provider of real-time, enterprise Web solutions, a contract related to its enterprise knowledge management and learning portal, Navy Knowledge Online (NKO). The contract includes additional scaling of NKO; ongoing support and maintenance of the portal; the purchase of additional hardware; beta-testing of

an at-sea version of NKO; and the development of Navy Knowledge Online SIPRNET (NKO-S) – a classified version of NKO. Additionally, NKO will be upgraded to Appian Enterprise v.3.0, the latest version of Appian's collaborative intranet solution suite.

Both the upgraded version of NKO and NKO-S will be deployed on BEA WebLogic Server.



www.appiancorp.com

Salesforce.com, BEA Announce WebLogic Workshop for sforce (San Francisco and San Jose, CA) – Salesforce.com, a leader in delivering software-as-a-service, and BEA Systems have announced that salesforce.com has joined the BEA Controls and Extensibility Program as a founding member and is developing an sforce-enhanced version of BEA WebLogic Workshop 8.1. Together, salesforce.com and BEA plan to deliver a co-branded bundled solution that will give developers pre-integrated access to all the capabilities in WebLogic Workshop and the sforce client/service application platform. BEA WebLogic Workshop for sforce will enable developers to integrate and distribute essential customer information throughout existing systems and enterprise infrastructure at a fraction of the usual time and expense.



www.sforce.com, www.bea.com

Toyota Australia Selects WebLogic Integration 8.1 for eBusiness Transformation Program

(San Jose, CA) – Toyota Australia has chosen BEA WebLogic Integration 8.1 as the software platform for its eBusiness Transformation Program, one of the largest information technology (IT)

integration projects ever undertaken by the company in Australia. The program, elements of which have already gone into production, is an initiative to integrate the systems and processes that connect Toyota and its trading partners to achieve real-time information access, enhanced two-way business visibility, and improved supply chain performance.



www.bea.com

BEA, Granite Systems Improve Telecom Italia's Performance (San Jose, CA and Manchester, NH) – BEA Systems, Inc., and Granite Systems, the communications software provider of service resource management solutions, have provided Telecom Italia, a European telecommunications carrier, with superior performance and significant ROI for its operational support systems.

Using BEA WebLogic Integration, used in Granite's Process Automator module, and Granite's Xng Core System, the companies developed a more efficient system designed to receive customer orders and perform automated design and assignment of Telecom Italia's circuits and services. BEA WebLogic Server was used as the system's underlying infrastructure. Once deployed, Telecom Italia was able to decrease its provisioning interval for DSL services by one-third, and reduce its operational costs 20%.



www.granite.com,
www.bea.com



Lowering the cost and complexity of developing secure, manageable enterprise-class Web services with BEA WebLogic Workshop™ and Confluent for BEA

With Confluent for BEA, Web services developed in BEA WebLogic Workshop™ 8.1 are automatically monitored, managed and secured at development time. Confluent's Control is bundled with WebLogic Workshop 8.1.



Real-time monitoring

No additional coding required, turn on instrumentation for a Web service, view all conversations as they execute, and drill down to views of individual operations and controls



Policy specification and enforcement

Define operational policies including security, logging and monitoring for Web services within BEA WebLogic Workshop 8.1



Explicit instrumentation with custom controls

Use Confluent Control to explicitly call operations such as monitoring and security from Web services

BENEFITS:

If you're a Developer—focus on application logic not infrastructure services

If you're an Architect—consistently implement security, change and other operational policies

If you're an Operations Manager—efficiently monitor, manage and evolve increasingly distributed applications

The screenshot displays the BEA WebLogic Workshop interface. The top window shows the 'Configure Policy' dialog for a web service, with a 'Request' section containing 'WSSecurityStep' and 'VerifyCertificateStep'. The bottom window shows the 'Runtime View' with a table of operations and a graph of operations called over time.

Operation	Date	Latency (ms)	Status	Policy
createOrder	Fri Jul 11 16:39:14, 2007	5047	Success	Success
queryStatus	Fri Jul 11 16:39:21, 2007	5047	Success	Success
cancelOrder	Fri Jul 11 16:39:22, 2007	5047	Success	Success

Policy specification and enforcements

Explicit instrumentation with custom control

Real-time monitoring



Confluent Software

Download Confluent for BEA at:
<http://www.confluentsoftware.com/solutions/download/beacib.html>

For more information on additional Confluent Web services management solutions, visit: <http://www.confluentsoftware.com>



Optimize J2EE.

The J2EE revolution is here to increase performance, value, and lower IT costs. It's a solution from Mercury Interactive that makes your whole J2EE applications ecosystem work right.

It's technology that tells you where to find the problems, easier, from development to the live applications. Even when deep within the source code.

It's about more than delivering J2EE applications. It's about delivering applications that work and yield real business value.

It's the Business Technology Optimization revolution.

And Mercury is the only one bringing you a revolutionary solution for J2EE.

Download our free white paper, "**Diagnosing J2EE Performance Problems Throughout the Application Lifecycle,**"

at www.mercuryinteractive.com/optimizej2ee

Get Optimized™

