

BEA WebLogic

DEVELOPER'S JOURNAL

SEPTEMBER 2002 - Volume:1 Issue:9

weblogicdevelopersjournal.com

eliminate IT chaos

BEA WebLogic Platform™ 7.0

dev2dev

FREE EVALUATION CD OF WEBLOGIC 7.0 PLATFORM

FREE EXPO PASS!
\$75 VALUE



Web Services Edge Exposition: San Jose, CA

A CONVERSATION WITH ADAM BOSWORTH



JP Morgenthal 32

FROM THE EDITOR
Good News for WebLogic
by Jason Westra
page 5

ADMIN CORNER
Recovering from an Invalid System Password
by Steve Mueller
page 36

NEWS & DEVELOPMENTS
Dirig Expands App Server Capabilities
page 50

WLS: Using an Implementation Model to Identify Packaging Issues 6
Classloader architecture in WebLogic Server
Andy Winskill

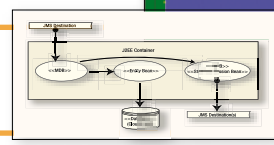


PERFORMANCE: JMS Performance Notes 12
The importance of testing your application
Peter Zadrozny

TRANSACTION MANAGEMENT: How Do They Go with the Flow? 18
There's more to business apps than app servers
Peter Holditch



EJB: The Promise of Entity Beans 22
A reality now
Vijay Mandava



INTEGRATION: Advanced JMS Design Patterns for WebLogic Server Environments PART 2 26
Using SonicMQ and the WLSA dapter
Hub Vandervoort & Jake Yara

PRODUCT REVIEW: JBuilder 7 Enterprise 40
From Borland Software Corporation
Andy Winskill



TIPS & TRICKS: WebLogic on the Mainframe 44
Jump-start your WebLogic deployment
Tad Stephens & Eric Gudjon



RETAILERS PLEASE DISPLAY UNTIL OCTOBER 31, 2002

\$15.00US \$16.00CAN



0 71486 03424 7 09>

BEA

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

Intel

www.intel.com/ad/bea

Wily Technology

www.wilytech.com

EDITORIAL ADVISORY BOARD

TYLER JEWELL, FLOYD MARINESCU,
SEAN RHODY

FOUNDING EDITOR

PETER ZADROZNY

EDITOR-IN-CHIEF

JASON WESTRA

EDITORIAL DIRECTOR

JEREMY GEELAN

EXECUTIVE EDITOR

GAIL SCHULTZ

MANAGING EDITOR

CHERYL VAN SISE

SENIOR EDITOR

M'LOU PINKHAM

EDITOR

NANCY VALENTINE

ASSOCIATE EDITORS

JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR

JENNIFER STILLEY

WRITERS IN THIS ISSUE

DAVE COOKE, ERIC GUDGION, PETER HOLDITCH,
VIJAY MANDAVA, JP MORGENTHAU, STEVE MUELLER,
TAD STEPHENS, HUB VANDERVOORT, JASON WESTRA,
ANDY WINSKILL, JAKE YARA, PETER ZADROZNY

SUBSCRIPTIONS

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department.

SUBSCRIPTION HOTLINE:

888-303-5282

Cover Price: \$15/Issue

Domestic: \$149/YR (12 Issues)

Canada/Mexico: \$169/YR

Overseas: \$179/YR

(U.S. Banks or Money Orders)

PUBLISHER, PRESIDENT, AND CEO

FUAT A. KIRCAALI

COO/CFO

MARK HARABEDIAN

VP, BUSINESS DEVELOPMENT

GRISHA DAVIDA

SENIOR VP, SALES & MARKETING

CARMEN GONZALEZ

VP, PRODUCTION & DESIGN

JIM MORGAN

ART DIRECTOR

ALEX BOTERO

ASSOCIATE ART DIRECTORS

AARATHI VENKATARAMAN

LOUIS CUFFARI • CATHRYN BURAK

RICHARD SILVERBERG

ASSISTANT ART DIRECTOR

TAMI BEATTY

VP, SALES & MARKETING

MILES SILVERMAN

ADVERTISING SALES DIRECTOR

ROBYN FORMA

ADVERTISING ACCOUNT MANAGER

MEGAN RING-MUSSA

ASSOCIATE SALES MANAGERS

CARRIE GEBERT • ALISA CATALANO

KRISTIN KUHNLE • LEAH HITTMAN

VP, SYS-CON EVENTS

CATHY WALTERS

CONFERENCE MANAGER

MICHAEL LYNCH

REGIONAL SALES MANAGERS

MICHAEL PESICK • RICHARD ANDERSON

ASSISTANT CONTROLLER

JUDITH CALNAN

ACCOUNTS PAYABLE

JOAN LAROSE

ACCOUNTS RECEIVABLE

KERRI VON ACHEN

ACCOUNTING CLERK

BETTY WHITE

WEBMASTER

ROBERT DIAMOND

WEB DESIGNERS

STEPHEN KILMURRAY • CHRISTOPHER CROCE

CATALIN STANCESCU

ONLINE EDITOR

LIN GOETZ

CUSTOMER SERVICE MANAGER

ANTHONY D. SPITZER

CUSTOMER SERVICE REPRESENTATIVE

MARGIE DOWNS

JDJ STORE MANAGER

RACHEL MCGOURAN

EDITORIAL OFFICES

SYS-CON Publications, Inc.

135 Chestnut Ridge Road, Montvale, NJ 07645

Telephone: 201 802-3000 Fax: 201 782-9637

SUBSCRIBE@SYS-CON.COM

BEA WebLogic Developer's Journal (ISSN# 1535-9581)

is published monthly (12 times a year)

Postmaster: Send Address Changes to

BEA WEBLOGIC DEVELOPER'S JOURNAL

SYS-CON Publications, Inc.

135 Chestnut Ridge Road, Montvale, NJ 07645

SYS-CON MEDIA

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



BY JASON WESTRA
EDITOR-IN-CHIEF

Good News for WebLogic

I have two newsworthy items to talk about this month. One concerns the application server market; the other pertains to a newly announced partnership in the wireless space. Each tidbit dates from July, but as editorial schedules run a bit behind the times, I'll relay them to you now.

In my last editorial, "The BEA Slayer?", I mentioned that free application servers would not affect BEA, whether they are open source like JBoss, or bundled with hardware like the Sun ONE Server and HP's Application Server. Interestingly enough, about two weeks after my editorial went to the printer, HP announced it was leaving the application server market! A Reuters report said, "HP, which has been losing money in software, said in a statement it would focus on areas where it has had market success and intellectual property.... It is pulling the plug on three middleware platforms used to glue networks together, including the Bluestone application server platform." No kidding. You don't make money on free stuff! On to wireless stuff...

Moving into my new home last month, I did my share to boost the economy by purchasing wireless networking equipment for my home office. I like to think of it more as an office "home" because wireless technology allows me to work anywhere in the house. I also had the opportunity to talk wireless with Scott Dietzen, CTO of BEA, just before BEA eWorld 2002. The hot topic Scott had for me was this new development tool called "Cajun," now known as WebLogic Workshop. Its promise is to make developing Web services easier for the average developer, who doesn't care about the gory details of J2EE, SOAP, and other Web services standards.

I had two burning questions to ask Scott. First, "With Workshop becoming the new IDE of choice for Web services on WebLogic, where does WebGain fit into the picture?" Well, a look on TheServerSide.com will tell you what's happening there.

Second, "Does BEA have any plans to incorporate wireless capabilities into WebLogic Server, or into Workshop?" WebLogic is already a great platform for WAP apps. Decks of WML cards can be dynamically generated, accessing back-end databases and middle-tier business logic. However, I want to see WYSIWYG WAP and J2ME (Java 2 Micro Edition) development. I want easy-to-use frameworks and the abstraction of standard data-synchronizing mechanisms like SyncML.

At the time, Scott was either not at liberty to talk about wireless partnerships, or he didn't anticipate how hot wireless vendors would be for Workshop. In any event, Research In Motion (RIM), a leading wireless company, and BEA announced their partnership to build an easy-to-use framework for delivering Web services to RIM's BlackBerry handhelds. The framework will integrate WebLogic Workshop and BlackBerry in a single environment, simplifying the development and testing of mobile apps.

Through their partnership, BEA and RIM are striving for an "always-on," "push" architecture for wireless applications. I have no idea what their plans from the handheld side will be, but my best guess is an all-Java solution. That means J2ME on the device. Although "push" isn't really a feature of J2ME 1.0, it could be added as a proprietary extension. It currently supports only a "pull" model, with the micro device polling for information at timed intervals.

No matter the application, WebLogic seems to be the solution, molding itself into the application infrastructure of today's global enterprises. Since the theme for this month's **WLDJ** is tools that make development and production environments easier, I hope you will also find tips and tools to make your current project(s) a breeze on WebLogic. Also this month, BEA debuts WebLogic Platform 7.0 with a full trial version CD included with this issue. See what thousands of development hours have produced. 🍷

AUTHOR BIO...

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

CONTACT: jason@sys-con.com

At Rosewood Software Services, we use Rational's Unified Process (RUP) and help many clients tailor it for their use. RUP specifies a number of models, including the Implementation Model, which is used to structure the physical artifacts (or files) within a project. For example, in a Java project the Implementation Model is used to define the project's Java package structure and the files used within the project, including class and JAR files as well as JSPs.

Most texts leave the Implementation Model at this point, using it as a simple model of file dependencies. The Implementation Model can be used to identify and design out additional issues, such as:

- J2EE packaging
- Identification of concurrent development opportunities
- Identification of COTS dependencies

- Application framework dependencies
- Testing strategies

This article explains how to use the Implementation Model to identify WebLogic Server (WLS) packaging issues.

Packaging Issues

The classloader architecture changed with WLS 6. This facilitated a number of new features, including support for EAR files and a much cleaner deployment model that supports hot deployment and hot redeployment.

CLASSLOADER BACKGROUND

A classloader allows a Java application to load new classes at runtime. Within a Java application, there is a hierarchy of classloaders.

The bootstrap classloader is the first in a sequence – the ancestor of all other classloaders. It's responsible for loading all of the JVM's internal classes as well as the classes in the java.* package. The extensions classloader loads all the JARs in the JDK's ext directory. The system classloader (often called the classpath classloader) loads all the classes in the classpath environment

variable or those specified using the `-classpath` command-line switch. The extensions classloader is a child of the bootstrap classloader and the system classloader is a child of the extensions classloader. Any classloaders created by an application (such as WLS) will be children of the system classloader.

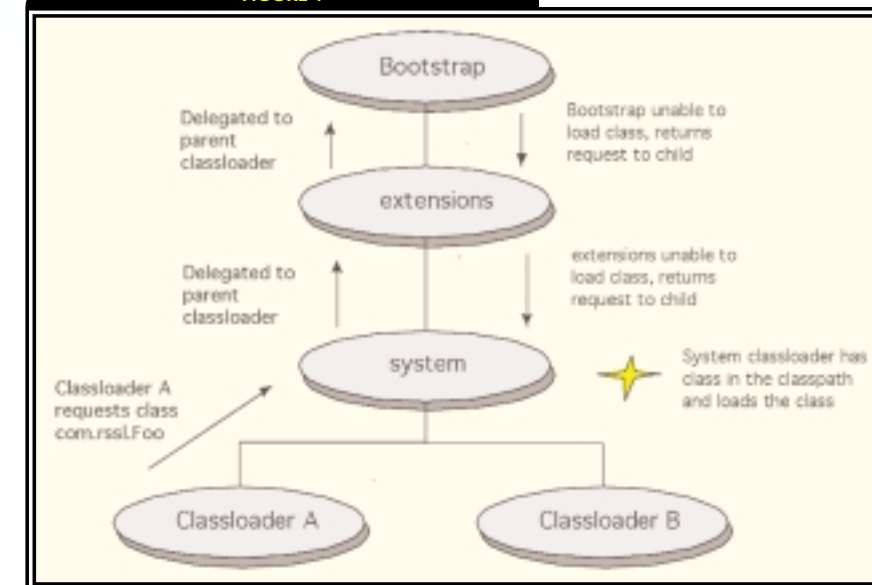
How is a class loaded? The classloader first checks to see if the class is already resident in memory. If the class isn't loaded, the classloader asks its parent to load the class. If the parent can't do this, the classloader attempts to load it itself. Since the classloader's parent is also a classloader, when it receives a request from a child to load a class it first asks its parent to load the class. This leads to requests being propagated up the classloader hierarchy to the bootstrap classloader. The bootstrap classloader then tries to load the file; if it can't, the request is passed down to the child that made the request. The request goes back down the hierarchy until a classloader is found that can load the class (see Figure 1). If the class can't be loaded by any classloader, a `ClassNotFoundException` is thrown.

To load and unload classes at runtime, WLS creates a new classloader for the application. In the context of WLS, an application is usually an EAR file, but it could also be a WAR or an EJB JAR file. When WLS creates a classloader for an application, it provides an effective separation of the applications – they can't access each other's classes due to the classloader delegation strategies.

In WLS, the application classloader loads all the classes explicitly associated with the EAR file and the classes in the EJB JAR files. WLS creates new classloaders for the Web

applications. In turn, these are responsible for loading any classes and libraries associated with the Web application. If a class is uniquely defined in a WAR file, it isn't accessible to other classloaders.

FIGURE 1



Standard Java classloader hierarchy

Using the Implementation Model to Identify Packaging Dependencies

In RUP, the Implementation Model is concerned with the structure of the physical directories and files that make up the application (or implementation subsystem – a collection of components). This is an important part of any development process; it's crucial to have a cohesive, decoupled Java package structure. Using the Implementation Model purely as a mechanism to identify the Java package structure means losing a vast number of modeling benefits. It can be used to identify additional issues, such as concurrent development opportunities and dependencies. For example, the Implementation Model can be used to help identify the packaging of EJBs.

If we can use the Implementation Model to identify packaging issues, can we use it to identify classloader issues? Indeed we can. Rational Rose helps with the automatic tracing of dependencies. If we take an example EJB, the EJB component may be in the Java package hierarchy `com.rssl.syscon.implmod.sampleEJB`. If the bean were dependent on another Java class, `com.rssl.syscon.implmod.util.ClassloaderDemo`, we'd expect to see a UML dependency (see Figure 2). This shows that the UML package `SampleEJB` is dependent on (requires access to) the `utils` package. In UML, a Java package is represented by a UML package.

Given the dependency shown in Figure 2, we can produce an EJB packaging diagram that reflects the new dependency (see Figure 3). Now a design decision should be resolved during the



BY
ANDY WINSKILL

AUTHOR BIO...

Andy Winskill is a principal consultant at Rosewood Software Services Ltd., UK. He specializes in BEA and Rational software and has more than 10 years of experience designing and constructing EAI and B2B applications. Rosewood Software Services is a BEA partner and a Rational partner.

CONTACT...

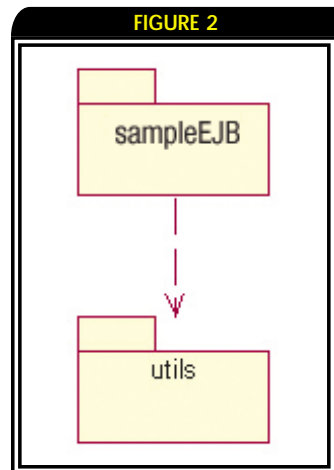
andy.winskill@rosewoodsoftware.com

Using an Implementation Model to Identify Packaging Issues

CLASSLOADER ARCHITECTURE IN WEBLOGIC SERVER

modeling activities: Where does the Classloader-Demo.class file reside? If there are no other dependents on this class, the class can be packaged with the EJB in its JAR file. If there are other dependents, where should it be located? The implementation-focused modeling activities should investigate these dependencies.

How do we identify classloader packaging problems? The issues presented by the classloader architecture are manifestations of dependency resolution. For example, if an EJB is dependent on class X and a Web application is also dependent on class X, there will be an issue if the class is physically different. A problem with models is that the pertinent information is dependent on the view the modeler takes during a particular activity. This often hides the information that may help identify packaging issues. It's helpful to promote the activity of "component dependency resolution," which generates a Packaging view within the Implementation Model (see Figure 4).

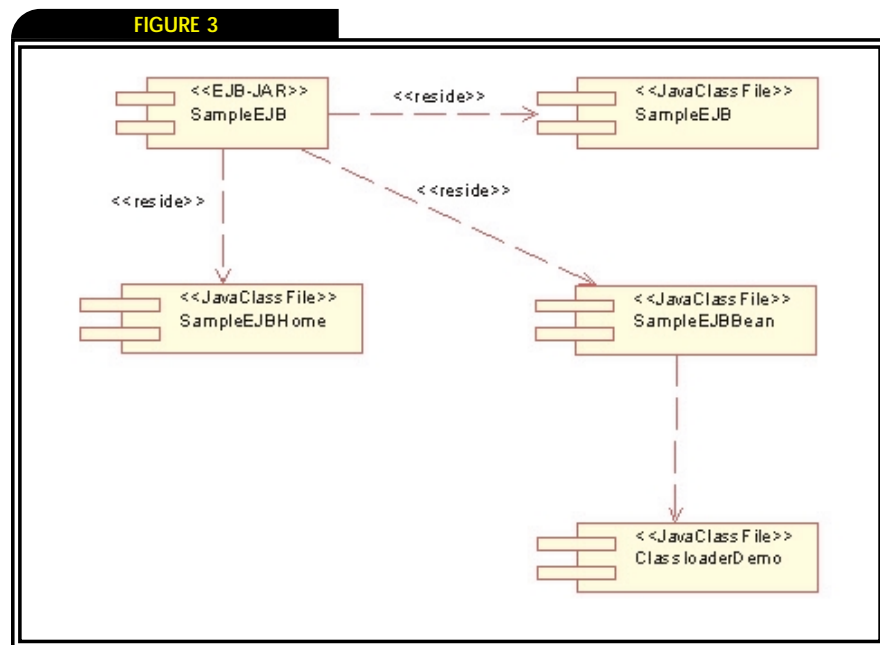


A simple package dependency

COMPONENT DEPENDENCY RESOLUTION

RUP is architecture-centric. It uses the 4+1 view of architecture, which has the Implementation view to focus on the architecturally significant structuring of the physical artifacts in the system. The Packaging view is a subset of the Implementation view and is architecturally significant in the J2EE environment due to the component-packaging dependencies that may impact how components can be distributed. To produce the view we have an activity that resolves the component dependencies, allowing us to correctly package the components.

The Component Dependency Resolution activity used in our development process (a cus-



An EJB with a class dependency

tomized version of RUP) focuses on resolving the dependencies in packaging and deploying components. We introduce a view in our Implementation Model that focuses on identifying the packaging our "build" will have and to provide a focus for the modeling investigation, although care should be taken in introducing unnecessary views as replacements for diagrams. This view is where we create the JAR/WAR/EAR components.

"The Implementation Model is an excellent artifact for investigating the architectural packaging dependencies that may be introduced during design"

We use Rational Rose to produce most of our models. Rose produces Java code in the Java package structure; in other words, it generates a UML package for each Java package. This package generation starts under the Component view (a high-level structure), under which we introduce the Implementation Model. This UML package is used to group all the views that relate to the implementation. Strictly speaking, it would be academically pure to have the top-level Java package under our Implementation Model package; unfortunately, this breaks the Rose code-generation facilities, which require that the top-level Java package be an immediate child of the Component view package.

Within our Implementation Model we introduce the Packaging Dependencies view, a stereotyped UML package. Here we investigate the dependencies between components and produce any relevant diagrams. The investigation starts with the creation of packages for each currently identified application or EAR file. This provides the top-level structure for the classloaders. From this we can partition the components between applications. In the example shown in Figure 4, our simple demo application has a single application classloader package. In large-scale applications where there may be multiple applications in the build, we would stereotype this package as <<Application Classloader>>. Within this package we produce diagrams to illustrate the contents the classloader should be responsible for loading.

Within our application loader package we

Dirig
www.dirig.com/illusion/weblogic

identify other significant packages, such as the Web applications that will be packaged into our application.

Where will there be conflict in the classloader dependency structures? If WebApplication1 has a dependency on the classes shown in Figure 5, there's a conflict with the dependents of SampleEJB. SampleEJB also requires access to the class file ClassloaderDemo. How will the class-

loader architecture resolve this? If the packaging remains and both the EJB and the Web application contain a copy of this class, the EJB class definition will be precedent. At this point we may want to investigate whether there's a flaw in the application architecture, or we may want to resolve the dependency.

Resolving component dependencies is usually a simple activity, and it's a vital design step. A rule of thumb: ensure that the class is loaded only once. Therefore, the dependency can be resolved by ensuring that a single classloader is responsible for loading it. This is achieved by putting the class at the highest common level – in this case, the demo application. If the class needs to span multiple applications (EARs), it must be located at the system level. At this point, questions should be asked of the architecture, and the issue of component versioning may appear.

An application is a discrete set of components. There shouldn't be dependencies between applications on class definitions. If there are, the class definitions have to reside at the system level. This implies that all applications within the application server will share the same version of the class. This may be the intent at the outset; however, when an application upgrade is required of a common class and this class is declared at the system scope, all applications must be recompiled and tested. In some organizations this may be difficult, if not impossible. A far better approach is to ensure that the application is cohesive and that corporate infrastructure, such as any corporate logging framework, is packaged along with the application. This way, modifications to other deployed applications won't have side effects. Should the Packaging view indicate a requirement to have classes at the system scope, it will be excellent documentation for that dependency.

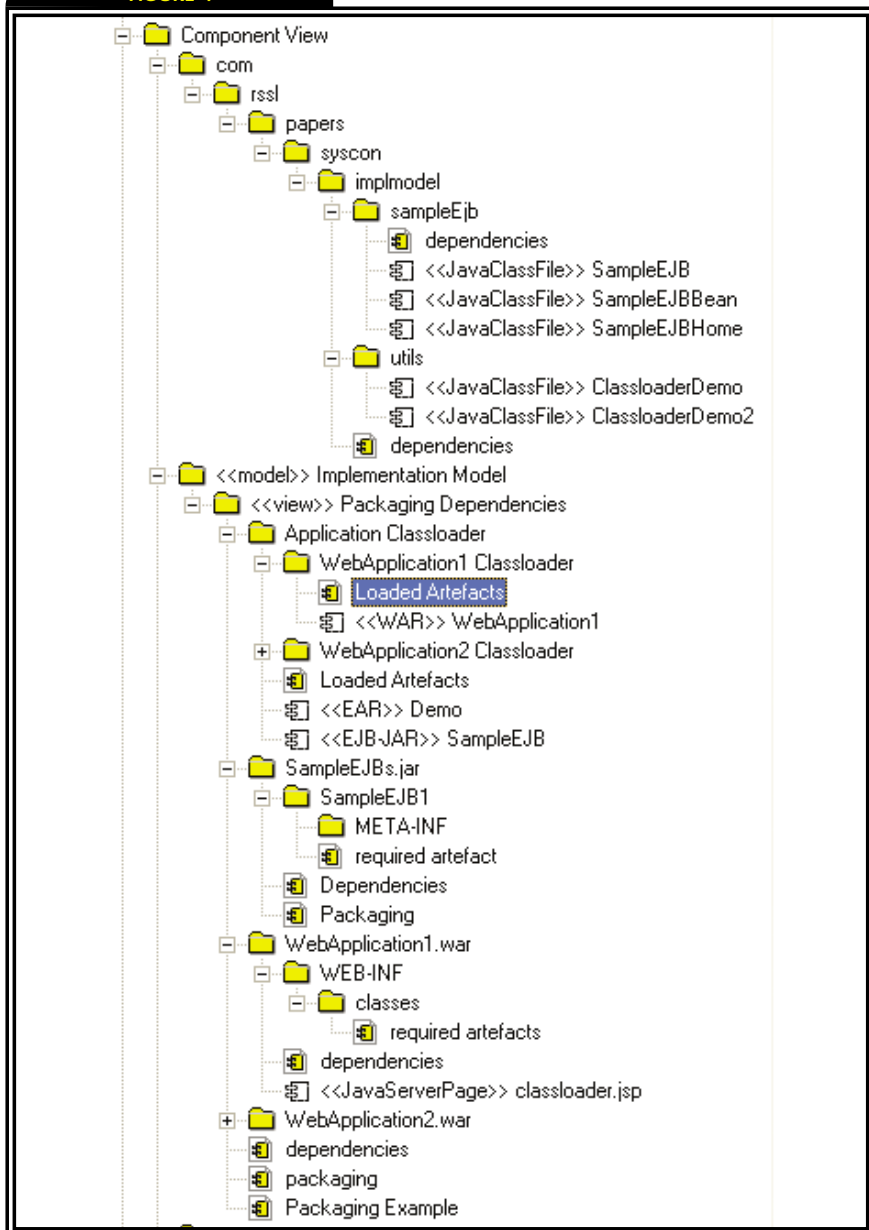
Summary

We've reviewed some of the issues and benefits of the classloader architecture in WebLogic Server. It's important to understand this architecture when architecting a WLS-based solution. The Implementation Model is an excellent artifact for investigating the architectural packaging dependencies that may be introduced during design but it shouldn't be generated just before cutting code – it should be used early in the development process. The Implementation Model may have significant impact on the application architecture you've chosen, and various techniques can be used within it to help identify the packaging dependencies.

Reference

- Kruchten, P. (1995). "The 4 + 1 View Model of Architecture." *IEEE Software*. IEEE. Vol. 12, issue 6.

FIGURE 4



Implementation Model with focus on the Packaging view

FIGURE 5



Dependency on these classes will cause a conflict

Rational Software

www.rational.com/offer/bea



JMS Performance Notes

THE IMPORTANCE OF TESTING YOUR APPLICATION



BY
PETER ZADROZNY

AUTHOR BIO...

Peter Zadrozny is the founding editor of *BEA WebLogic Developer's Journal* and the chief technologist for BEA Systems in Europe, the Middle East, and Africa.

CONTACT...

z@bea.com

It's almost impossible to address the performance of the JMS implementation on the WebLogic Server in a generic fashion. Message size, acknowledge mode, persistence mode, and type of consumer are just a few of the things that can impact the performance. Add the JVM, the operating system, and hardware, which can also affect the performance, and you begin to see why we can't generalize. With so many variables, it's not possible to extrapolate the performance of another JMS-based application, no matter how similar to yours it seems. The only way to understand JMS performance is by testing your own application (or a proof of concept).

It's possible, however, to run a few tests to see the cost differentials of using various options, thus setting a level of expectations for the general behavior of JMS. Since we can't test all possible combinations, we'll limit ourselves to a subset of options using a simple application, a stock ticker. This is the most popular example of publish-and-subscribe messaging.

A stock ticker works by continuously presenting all the trades that occur in a stock exchange, providing the name of the company, the number of shares traded, and the price of the trade. This can be cumbersome for those only interested in the trades of a few companies. Using the Pub/Sub messaging model, they can view only the trades of the companies of interest.

From the JMS perspective, a stock ticker application can be seen as one that sends an event to a topic for every trade that occurs in a stock exchange. Consumers receive only the desired events from selected companies by subscribing to the appropriate topic and specifying the desired companies. In this case, the message selector is the company name or symbol. Before going into more detail on the test application, let's review how performance is measured.

A common problem associated with performance testing a messaging-based system is misunderstanding the performance metrics. Performance of asynchronous messaging systems is typically measured based on throughput. In this case, the most obvious throughput measurement is "messages per second" (MPS). However, you have to be very careful with this measurement because throughput is a measure of capacity, not speed. MPS tends to be interpreted as a measurement of speed, which is not the case. Consider, for example, consumers who can't process messages fast enough – the messages are just waiting in the corresponding queue or topic. Alternatively, when the message publishers can't produce a high enough rate of messages, the

consumers are just waiting. In both examples, the messaging system handles messages at a pace imposed by external factors (message producers and consumers). It's important to measure the throughput for both the message producer(s) and the consumer(s) because each is heavily dependent on the other.

The Stock Ticker Application

This example uses an oversimplified version of a rather primitive stock ticker application where there's only one producer or publisher of trades in the stock exchange. Each trade is a message placed on a single topic. The single publisher continuously publishes events to the topic. The event in this case is a message of a particular

type identified by a property in the message, which contains the symbol of the company for which the trade was done. There is one trade per company, and each company on the exchange trades in an orderly, sequential fashion (this is extremely simplistic, but still effective for testing purposes).

On the consumer side, each customer subscribes to a few of the companies that trade in the stock exchange. Some of the message consumers subscribe to listen to events (trades) of the same companies, so some subscribers will receive the same messages.

Note that because of the transient nature of the information, it doesn't make sense to have durable clients or persistent messaging. If the message detailing a particular trade is lost for any reason, by the time it's recovered from the server it will probably be obsolete because many other stock trades will have occurred in the meantime. In this kind of application it makes more sense for the subscriber to simply wait for the next message. With this in mind, our tests are limited to only two acknowledge modes: NONE and MULTICAST with no persistence (bear in mind that this refers to the subscriber; the publisher acknowledgement mode is always AUTO_ACKNOWLEDGE).

Testing Environment

These tests use WebLogic Server 6.1 SP2, JDK 1.3.1-b24 with a heap of 256MB running on a Sun Ultra 60 (dual Ultra SPARC 450MHz, 512 MB of memory). The load is generated using The Grinder (<http://grinder.sf.net>; see the related article in *WLDJ*, Vol. 1, issue 7). There are special plug-ins for the functionality of the stock ticker publisher and the consumers.

In this example, the publisher creates 100 different types of messages (0-99), each 64 bytes, which is the approximate size of this kind of message. The stock exchange consists of 100 companies, where each company makes one trade at a time, always in a sequential fashion.

Think time isn't used for publishing the messages in these tests. However, the publisher does write a line to The Grinder log file for every message published, which makes the simulation more realistic.

On the other side of the messaging system, the subscriber plug-in simulates, in a very basic way, a trader that is subscribed to receive events of 25 of the possible 100 companies. Every trader runs on its own JVM and establishes a JMS connection and session before it starts the test run. During

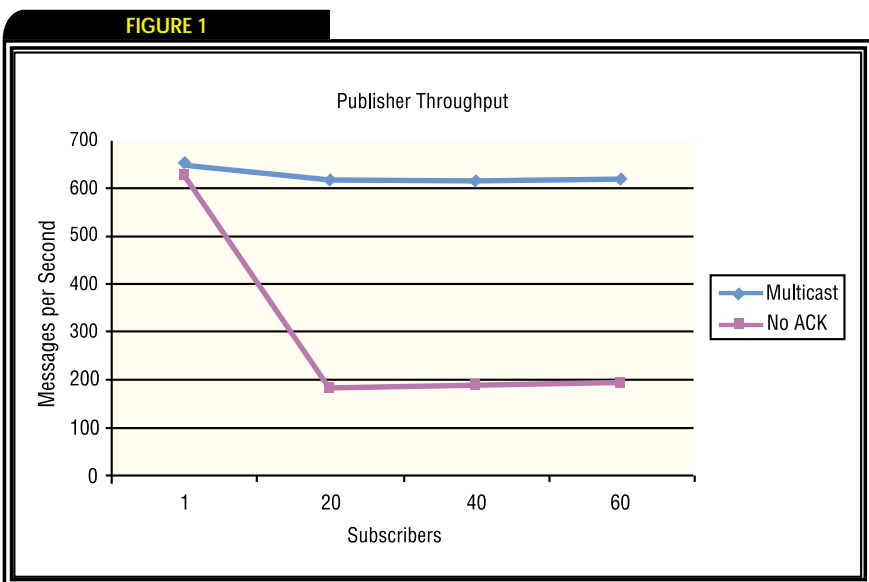
TABLE 1

SUBSCRIBER LOAD	1	20	40	60
Publisher	627.50	184.20	189.50	195.10
Subscriber	156.40	25.77	25.50	25.27

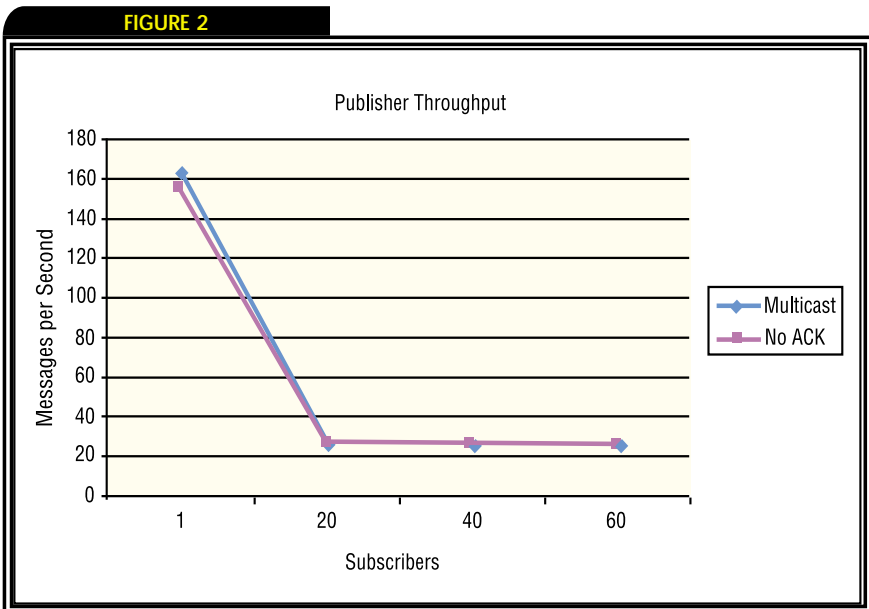
Publisher and subscriber performance

the test run it receives the messages it's subscribed to, in this case a range of 25 contiguous message types where the first type has been selected randomly. In real life, every trader would have subscribed to a number of companies that aren't likely to be a block of an alphabetically ordered list of companies; it's modeled this way for the sake of simplicity.

"The only way to understand JMS performance is by testing our own application"



Results of publisher throughput



Subscriber performance

The subscriber does nothing but write a line to The Grinder log file for every message received. This is very important because it has an impact on performance, and your application is likely to do an operation as time-consuming as writing to a file.

The publisher and each consumer run on their own JVMs on four computers (Pentium III 600MHz, 256MB memory, SuSE Linux 7.0). Special care has been taken to ensure that no paging or swapping occurs during the execution of the test runs.

Test Runs

Publisher and subscriber performance (MPS) are investigated using the no-acknowledgement mode under various subscriber loads (see Table 1).

- For the case of one subscriber, the publisher: subscriber performance exhibits a 1:1 relationship (remember that the consumer is subscribed to only 25% of the messages).
- For other subscriber loads, the throughput is very stable, at around 25 MPS.
- For other subscriber loads, the 1:1 relationship no longer exists. This happens because the consumers are waiting for the events they're subscribed to. If every consumer were subscribed to exactly the same block of companies, we'd expect to see a rate of about 47 MPS, but the beginning of the block is randomly selected. This means many consumers will be idle waiting for the first message of their block. This idle time decreases the average of messages received.

We repeated the above test run using the multicast no acknowledge mode. We expected this mode to be faster, but it's less reliable. Figure 1 compares the results of the publisher throughput for the two modes.

As expected, the throughput is substantially faster using multicast – a little more than three times faster. Figure 2 depicts the same comparison for subscriber performance.

Since we don't observe a similar trend for the subscriber performance, we must investigate why. First, we rule out the possibility that we're losing messages by looking at the network usage for 60 consumers (see Figure 3).

With a paltry 3.5% utilization, it's hard to imagine that messages are being lost because of high traffic. This is interesting because the network utilization here is less than one-half of that observed in the no acknowledge mode. We then check the CPU usage of the computer running the JMS topic in Figure 4.

Again, the activity is about half that observed using the no acknowledge mode. Thus, we're convinced we're not losing messages for these reasons.

Precise Software

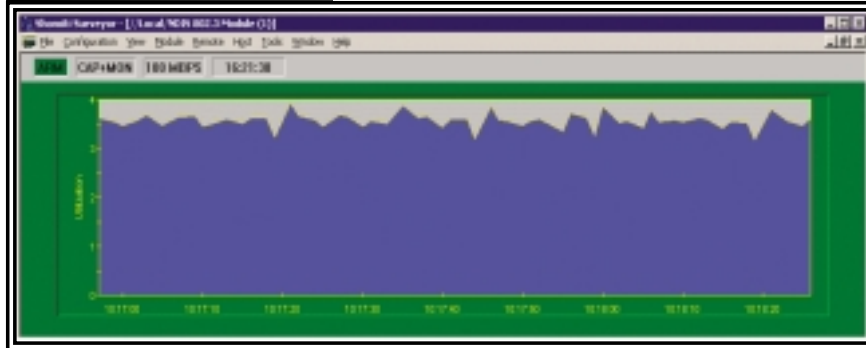
www.precise.com/wldj

TABLE 2

SUBSCRIBER LOAD	1	20	40	60
Publisher	188462	52322	55535	58708
Subscriber - actual	47006	7744	7699	7603
Subscriber - expected	47116	13081	13884	14677
Differential	0%	41%	45%	48%

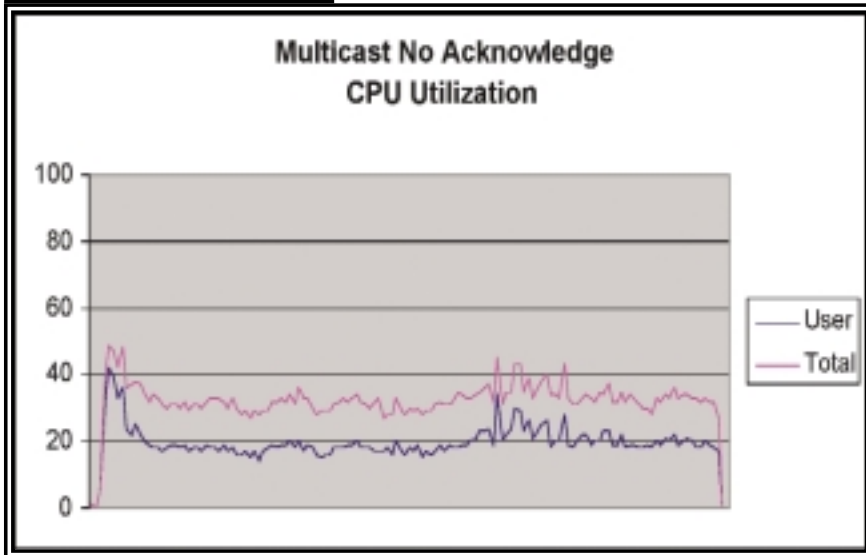
Actual number of messages

FIGURE 3



Network usage for 60 consumers

FIGURE 4



CPU usage

TABLE 3

SUBSCRIBER LOAD	1	20	40	60
Publisher	193044	185146	184173	185363
Subscriber - actual	48308	7696	7535	7605
Subscriber - expected	48266	46287	46043	46341
Differential	0%	83%	84%	84%

No acknowledge test runs

TABLE 4

60 Subscribers - 2 ms TT	No ACK	Multicast
Producer TPS	49.69	49.81
Consumer TPS	12.46	12.26

Test runs with a sleep time of 2 milliseconds

Custom Grinder plug-ins provide us with the actual number of messages handled during the sample period, so we can proceed to analyze this. First, we examine the number of messages handled using no acknowledge mode. Table 2 shows the actual number of messages produced by the publisher and received by the consumer. Our expectation that each subscriber could handle, in the best-case scenario, 25% of the published messages isn't possible because of the overlap between the various blocks of companies to which the consumers are subscribed. Thus, the differential of 40–50% seems reasonable. The differential increases as the number of consumers increases, which again seems reasonable. Next we perform the same analysis for the multicast no acknowledge test runs (see Table 3). The differential is almost double the expected 40–50%.

There are a few things happening here. The messages aren't getting lost; they're in the topic. We proved this by stopping the producer of the messages; after a few minutes the consumer had picked up all the messages. More important, the consumers are already at their maximum speed; changing the transport mechanism from no ack to multicast no ack will not make things go faster.

Using an analogy, with the no acknowledge mode we're drinking water from a glass. With the multicast no acknowledge mode we're drinking water from a fireman's hose. A couple of test runs with 60 consumers illustrate this, this time using a sleep time of 2 milliseconds before publishing every message (see Table 4).

As you can see, messages are now published and consumed at about the same rate. This example illustrates that:

- MPS is more a measure of throughput capacity than plain speed. Notice how the publisher MPS decreases from 195 MPS for no acknowledge mode to about 50 MPS with the addition of a 2-millisecond sleep time before publishing every message.
- You have to be very careful when defining the throughput for your application and interpreting the results.

Conclusion

No matter how similar your application might look to another, you can't extrapolate performance results. Testing your application is the only way to really understand JMS performance.

Acknowledgements

This article is an extract from the book *J2EE Performance Testing* by Peter Zadrozny (Expert Press, June 2002). Thanks to Phil Aston for writing the custom plug-ins for The Grinder. The software used in these tests can be downloaded from www.expert-press.com.

Panacya
www.panacya.com



As I may have mentioned once or twice in this column over the foregoing months, developers can derive a large amount of value from building their applications on an application server.

Transactions: How Do They Go with the Flow?

THERE'S MORE TO BUSINESS APPLICATIONS THAN APPLICATION SERVERS

BY PETER HOLDITCH



AUTHOR BIO

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

CONTACT...

Peter.Holditch@bea.com

Services accessed by standard APIs, such as transaction services provided through JTA and the services of the EJB container, take a lot of the technical grunt work out of transforming business requirements into deployable applications and allow developers to devote more attention to the business logic, and less to the technical niceties of how this logic should be realized in a physical implementation.

There is more to business applications than application servers, however. The J2EE APIs (with the exception of JMS) tend to be focused around a synchronous programming model where a request for information (often from a Web browser) arrives, some code runs to execute whatever business logic there might be, and a response is sent back to the client. This model is fine for simple applications, but in real life, the front end that a user wishes to interact with is likely to be an aggregation of display elements for many back-end systems, and many business processes may involve interaction with partners, or offline systems, or they may simply run for a long time – where long is defined as longer than a user would wish to wait at his desk

for a response to come back from the system.

To take care of these more complex – and more realistic – scenarios, BEA provides the WebLogic Platform – a higher-level set of platform services layered over the application server to deliver support for long-running business processes and back-end integration in the WebLogic Integration component, and to aggregate and display content at the front end in the WebLogic Portal.

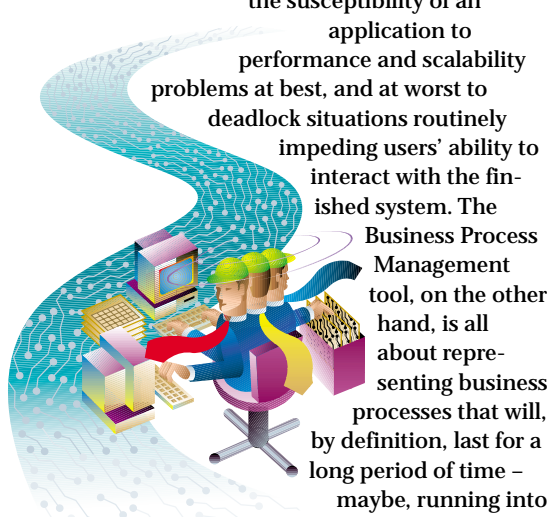
Completing the picture, WebLogic Workshop provides a runtime framework to abstract the complexity of using the synchronous J2EE programming model for asynchronous Web service implementations.

That's all well and good, but what does it have to do with transactions? In this column I'll look at the WebLogic Integration Business Process Manager (BPM) and how it uses the JTA subsystem provided by the underlying application server to bring ACID transaction properties to bear on potentially long-running business processes. This will not only serve as an introduction to one feature of the BPM tool itself, but give you some indication as to how the raw J2EE programming model in general, and JTA in particular, could be applied in a real design to a real business process.

So, What's the Problem?

As I've discussed here several times before, JTA transactions should be kept as short as possible to avoid locking a large quantity of distributed data for long periods of time, thereby increasing the susceptibility of an application to performance and scalability problems at best, and at worst to deadlock situations routinely impeding users' ability to interact with the finished system. The Business Process Management tool, on the other hand, is all about representing business processes that will, by definition, last for a long period of time – maybe, running into multiple years.

Given the need to track process instances over these long periods of time, a database is clearly



Covasoft

www.covasoft.com/today's_tech



going to be necessary to store the state of all active workflow instances – try keeping in-memory state information reliably for a period of years.... Also, given that this database will be transactional, combining this representation of the process's state transactionally with the execution of the individual steps that make up the overall process will enable JTA to deliver once and once-only execution of each step in the process and minimize or avoid the need for compensating workflows and other such techniques to back out changes in the case of a failure. This, after all, is exactly what JTA does best. However, a trivial mapping of one workflow instance to one JTA transaction is clearly not what we want. We have already said that a workflow instance could be active for years, and also that a JTA transaction should be active for as short a time as possible, two obviously conflicting goals.

“The Business Process Management tool...is all about representing business processes that will by definition last for a long period of time”

Workflow: Marriage While You Sleep

So how do you design a marriage between the close-coupled JTA transaction on the one hand, and the long-running workflow on the other? To find the answer, we need to discover why workflows can run for such a long time. They clearly aren't just hanging around for the sake of it. The reason always turns out to be that they're waiting for something, usually an event occurring in an external system (maybe even a system in someone else's organization). While an instance of a workflow is waiting for something to happen in this way, it no longer needs to be in the app server's memory executing. It can be swapped out to the database and the workflow engine can commence processing other useful work that can execute. The instance is said to be quiesced. You will have spotted that swapping the instance state to the database required a database update, which in turn will require the transaction to be


committed. Yes, you've guessed it – the workflow engine will run tasks up to the point where an instance has to quiesce within the context of a single JTA transaction, and before it quiesces, it will commit the transaction. In fact, that's not strictly accurate. If a workflow instance was invoked programmatically within the context of an existing JTA transaction, then the workflow engine won't try to commit the transaction; it will merely perform the database update and return control to the caller.

In general, the workflow engine will start a transaction when an instance is activated (unless one is already in force, started externally) and execute actions as instructed by the workflow (including actions in and called sub workflows) up until the point when a quiescent state is reached. Then the state will be written away to the database and the transaction will be considered complete from the workflow engine's point of view. Workflow actions themselves (for example, call a business operation implemented in an EJB, call an application through an adapter) will also be invoked in the context of the transaction (subject to the usual rules enforced by the containers as defined in deployment descriptors). Provided they can support XA transactions, any updates that they cause to happen will commit or abort atomically with the progress of the workflow instance itself giving the desired once and once-only semantics that can so simplify coding of business logic.

Two final points: first, what happens when things go wrong? The workflow engine provides an exception-handling capability, and it is the responsibility of the active exception handler to decide if the exception is recoverable and commit, abort, or mark abort-only the transaction as appropriate. Second, what if I want to explicitly break up one workflow that just consists of a set of synchronous calls into multiple transactions? To achieve this, you must force the workflow instance into a quiescent state. This can be done by inserting an event node that triggers itself, assigning a task to a user or various other tricks documented in the manual. Thus, if you want transaction granularity smaller than the default, you can achieve it.

That's it for this month, I'll flow off now and quiesce for a while. ZZzzzz...

References

A short article necessarily can't cover every detail of such a wide-ranging topic as workflow and transactions. For the complete description of the transactional behavior of the WebLogic Integration Business Process Management engine, refer to chapter 7, “Understanding the BPM Transaction Model,” in the “Programming BPM Client Applications” manual. 

Sitraka JClass ServerViews

www.sitraka.com/jclass/wldj

The Promise of Entity Beans

A REALITY NOW



BY VIJAY MANDAVA

AUTHOR BIO...

Vijay Mandava is a technical manager in professional services at BEA. Vijay has 10 years of experience designing and constructing large-scale, object-oriented distributed systems. He has bachelor's and master's degrees in computer science.

CONTACT...

vijay.mandava@bea.com

Working as a BEA consultant, I've helped customers successfully design and deploy applications on various versions of the WebLogic Server (WLS). BEA has been supporting container-managed persistence (CMP) entity beans since EJB 1.0, and a few of our customers have used them. Unfortunately, some used them without understanding the ramifications; others heard about performance constraints and completely excluded entity beans from their architecture/design choices.

It was difficult to come up with reasons to use CMP entity beans when chief architects asked, "When should I consider using CMP entity beans at all?" The nirvana of entity beans lies in two things: caching and object-relational mapping. Both were done poorly in the earlier implementations, but EJB 2.0 came to the rescue. WLS 7.0's add-on features to the EJB 2.0 specification

bridge the gap in performance between using entity beans and using stateless session beans with DAO and JDBC – in some cases even performing faster than the latter. When using JDBC, developers are strongly advised to use container-managed transactions.

Problems with Earlier Versions

Several customers who used entity beans in the past had to redesign and use JDBC because their systems were too slow to meet service-level agreements. That slowness was due to some of the following problems:

- **No caching:** Each transaction went to the database to load the entity bean. One of the promises customers look for in using entity beans is caching, and none was available for regular entity beans in a clustered environment.
- **Single-table support (O-R mapping too simplistic):** Customers were also looking for object-relational mapping to be done by the entity bean, but the mapping provided by the container was too simplistic.

- **Single instance of the entity bean per virtual machine (VM):** Because WebLogic 5.1 and below only supported exclusive locking, all the calls to the entity bean were serialized, causing bottlenecks even for read-only beans.
- **Exclusive locking:** Deadlocks were highly likely when some beans used transactions and some did not.
- **Single setX call:** This caused the entire entity bean to be written to the database. Before the EJB 2.0 specification, the container didn't have any hooks for determining what changed and what didn't. This meant that on an ejbStore, all the attributes were updated.
- **Loading:** Loading the entity bean caused all data members to be loaded in memory, even if only a couple were needed.
- **No dynamic queries:** EJB had to be redeployed to write a new query.
- **Queries:** Queries can instantiate large numbers of entity beans, consuming memory and degrading performance.

WLS 7.0 Features

WLS 7.0 implements the EJB 2.0 specification, providing for richer object-relational mapping. EJB 2.0 also provides the container with a lot more flexibility for doing optimized reads and writes to the database. The following features elucidate the richness and flexibility developers have in designing entity beans in WLS 7.0.

- **Container-managed relationships:** Entity beans can have relationships with other beans; these relationships can be either bidirectional or unidirectional. WLS supports three types of relationship mappings managed by WebLogic CMP: one-to-one, one-to-many, and many-to-many.
- **Multiple table mapping to an entity bean:** Multiple table support allows an implementer

of EJB 2.0 CMP beans to map a single EJB to multiple DBMS tables within a single database.

- **Tuned writes with EJB 2.0:** Instead of writing all the fields to the database when ejbStore is called, only the updated fields are written to the database.
- **Tuned reads using field groups:** On an ejbLoad the container, instead of loading all the fields of the entity bean, loads only fields in the field-group.
- **Caching of relationships:** This feature increases entity bean performance by loading related

beans into the cache in a single join query instead of multiple queries.

- **Entity beans can return ResultSets:** WLS supports ejbSelect() queries that return the results of multicolumn queries in the form of a java.sql.ResultSet.
- **Application-wide cache for heterogeneous entity beans:** Instead of a separate cache for each entity bean, this feature enables multiple entity beans within an EAR to share a single runtime cache.
- **Dynamic queries:** The EJB 2.0 specification forces users to hard-code queries in the deployment descriptor. With dynamic queries, new queries can be constructed and executed programmatically without redeploying the beans.
- **Read-mostly design pattern:** In the real world, most applications are 90% read and 10% update. An entity bean can be written to model the data and can be deployed as both a read-only entity bean and a regular entity bean. With this approach, users who want to read data talk to the read-only entity bean, and users who want to update data talk to the regular entity bean. When an update occurs, the container invalidates all the read-only entity beans, forcing the container to call ejbLoad the next time the data is accessed.

Other useful features include automatic primary key generation, cascade delete support, EJB QL support, ejbSelect methods, and the concurrency strategies described below.

CONCURRENCY STRATEGIES

A concurrency strategy defines how many instances of an entity bean are created, who does the locking to maintain transactional integrity, and the access pattern of the data modeled in the entity bean. The right concurrency strategy can

make a tremendous difference in the performance of your entity beans.

- **Exclusive:** The container creates only one instance of an entity bean per VM, and all calls to the entity bean are serialized as the container locks the entity bean on use (for both read and write). This is never recommended.
- **Database:** The container defers locking to the database, and each transaction gets its own copy of the entity bean. `ejbLoad` and `ejbStore` are called at the beginning and end of the transaction respectively.
- **Read-only entity beans:** Neither the database nor the container holds any locks, and each transaction gets its own copy of the entity bean. A configurable parameter called `<read-timeout-seconds>` controls when `ejbLoad` is called on the entity bean. `ejbStore` is never called on the entity bean. Clients can still call create, remove, and update operations on the entity beans. The creation and removal will be successful, but the updates won't modify the database because `ejbStore` isn't called. It's the programmer's responsibility to not call CUD (create, update, delete) methods on read-only entity beans. Read-only entity beans are a perfect solution to the "distributed cache" problem.
- **Optimistic entity beans:** The container defers locking to the database, but locks aren't held during the transaction. The basic idea is that the container checks for modified data before committing and rolls back if someone else has modified it. This is useful if you want higher consistency than `TX_READ_COMMITTED` but don't want to go as high as `TX_SERIALIZABLE`. Use it if you can live with reading stale data for a short time but want complete transactional integrity for updates. There are four options for checking optimistic conflicts:
 1. Check columns that were read
 2. Check columns that were modified
 3. Check timestamp column
 4. Check a version column

Options 3 and 4 aren't recommended because the schema of the table needs to be changed to incorporate this concurrency strategy.

Also, with the optimistic concurrency strategy, you can configure whether you want caching between transactions to be true or false. With caching between transactions set to false, `ejbLoad` will be called at the beginning of each transaction.

Comparison

With CMP, there will always be additional processing due to the integration of the EJB container and a layer of container-generated code-handling transactions, security, pooling, life-cycle management, failover, caching, and relationships. CMP (by design or by spec) has to do internal operations, (`ejbLoad`, `ejbStore`, `ejbActivate`, `ejbPassivate`) as opposed to JDBC logic manually programmed by developers. The benefits of the container infrastructure are optimized, generated database access; accelerated development; and simplified code maintenance.

On the benchmarks I've seen, unless the data is cached, [stateless session bean + DAO (doing JDBC)] has performed 30-50% faster than the entity bean implementation.

When to Use What

Entity beans shouldn't be used as a substitute for writing JDBC. Use entity beans if your object-relational model isn't overly complex (involving numerous joins, etc.) and flexibility and code maintenance are more important than raw speed for your project.

Use JDBC for simplistic, atomic-blind updates; to integrate with stored procedures and triggers; and to handle large ResultSets.

WebLogic add-ons such as the read-mostly design pattern and optimistic caching with cache-between-transactions set to true are two design choices that make entity beans an attractive option. Both are BEA-proprietary, and both are specified in the WebLogic-specific deployment descriptor. No code changes are needed when migrating to another J2EE-compliant application server. Use optimistic caching concurrency strategy if it's OK for the application to read stale data for a short period of time.

Use the read-mostly design pattern for the use case in which you read most of the time and update less frequently. This design pattern also has the shortcoming that the readers can read stale data. In case some of the readers should not read stale data at all, they can be made to read from the read-write bean.

Read-only entity beans for CMP entity beans are mentioned in the features deferred to future releases of the EJB specification. Use read-only entity beans to implement a distributed cache that can be refreshed periodically.

Conclusion

Most real-world applications do far more reads than writes. Implementing the read-mostly pattern will provide the best of both worlds. It provides easy development and flexible deployment, along with excellent performance characteristics for accessing data and paying extra overhead only when data is being modified. Optimistic concurrency with cache-between-transactions set to true can be faster than JDBC. Writing optimized SQL is hard for regular Java programmers, so don't ignore entity beans – evaluate and determine whether they're a good fit.

Entity Beans Examples

Examples are provided in subdirectories of `BEA-HOME/samples/server/src/ejb20`.

- For an example that demonstrates the relationships, look at the `BEA-HOME/samples/server/src/ejb20/relationships/bands` directory.
- For an example that demonstrates multiple table mapping, look at the `BEA-HOME/samples/server/src/ejb20/multitable` directory.
- To use the database concurrency option specify `<concurrency-strategy>Database</concurrency-strategy>` in the `weblogic-ejb-jar.xml`.
- To use the read-only concurrency option, specify `<concurrency-strategy>Read-Only</concurrency-strategy>` in the `weblogic-ejb-jar.xml`.
- To use the optimistic concurrency with cache-between-transactions and check on Modified option specify `<concurrency-strategy>Optimistic</concurrency-strategy>` `<cache-between-transactions>True</cache-between-transactions>` in the `weblogic-ejb-jar.xml` and specify `<verify-columns>Modified</verify-columns>` in the `weblogic-cmp-rdbms-jar.xml`.
- To implement the "read-mostly" design pattern register the bean implementation as two EJBs, one as read-only and the other as Read-Write, and add the `<invalidate>ejbname</invalidate>` where `ejbname` is the read-only entity bean name in the `weblogic-ejb-jar.xml`.

References

- <http://edocs.bea.com/wls/docs70/ejb/index.html>
- <http://java.sun.com/products/ejb/docs.html>

Sitraka PerformaSure

www.sitraka.com/performance/wldj



BY HUB VANDERVOORT
& JAKE YARA

AUTHOR BIOS...

Hub Vandervoort is vice president of Professional Services for Sonic Software. He has more than 20 years of experience as a consultant and senior technology executive in the networking, communications software, and Internet industries.

Jake Yara is an independent software consultant working with Sonic Software to implement application server-specific solutions for the leading EJB application servers on the market. In February 2000 he founded Yara, Inc.

CONTACT...

hvanderv@sonicsoftware.com
jakeyara@alum.mit.edu

Advanced JMS Design Patterns for WebLogic Server Environments

PART 2

USING SONICMQ AND THE WLSADAPTER

Part 1 of this article (WLDJ, Vol. 1, issue 8) explored third-party Java Message Service (JMS) integration into WebLogic Server (WLS) and addressed related issues. In Part 2, we'll implement transactional JMS design patterns using

SonicMQ and the WebLogic Server Adapter (WLSAdapter) as the JMS solution. Included in this discussion are the message-driven bean (MDB), message-producing bean (MPB), and message-consumer bean (MCB).

Pattern 1: Message-Driven Bean with a Queue Listener

This design pattern is used with the WLSAdapter to create and configure an MDB that will be used with a SonicMQ Queue to implement an XA transaction. The destinations and connection factories will be stored in a file-based JNDI store.

CONNECTION FACTORY

The first step is to create the JNDI entry for the connection factory. This can be done with a JNDI loader application that operates outside of WebLogic, or in a startup class that creates the entries within the WLS internal JNDI store (see end of article). We'll assume a file-based JNDI store for this example. Use the WLSAdapter to create the XAConnectionFactory.

```
com.sonicsw.sonicmq.j2ee.wls.XAQueueConnectionFactory
xafactory =
new com.sonicsw.sonicmq.j2ee.wls.XAQueueConnection
Factory
("localhost:2506", "QueueMessageDriven", "mypass
word");
```

This creates a connection factory named "xafactory" that's bound to a JNDI entry. The variable `m_context` is a reference the file-based JNDI context created from the `com.sun.jndi.fscontext.ReffSContextFactory` package.

```
m_context.rebind("myJMSxaqcf", xafactory);
```

At this point, a connection factory named "myJMSxaqcf" is ready for use in the JNDI store. A reference to the queue in the JNDI store is now created:

```
javax.jms.Queue queue = new
progress.message.jclient.Queue
("sonicQueue");
```

This queue would also be bound to a JNDI entry:

```
m_context.rebind("sonicQueue", queue);
```

DEPLOYMENT DESCRIPTORS

An EJB deployed in WLS requires two descriptor files: a standard "ejb-jar.xml" and a WebLogic-specific file, "weblogic-ejb.xml". In `ejb-jar.xml`, the MDB is prepared for an XA transaction by setting the `trans-attribute` tag to "Required" (see Listing 1; the listings for this article may be found online at www.sys-con.com/weblogic/sourceec.cfm).

The queue and connection factory are designated in the WebLogic-specific descriptor. The tag `destination-jndi-name` identifies the destination, in this case the queue "sonicQueue". The connection factory "myJMSxaqcf" is identified in `connection-factory-jndi-name` tag (see Listing 2).

MESSAGE-DRIVEN BEAN

The MDB doesn't require anything special to actually receive the message in an XA transaction. It just needs to be deployed transactionally in WebLogic, and it must implement the `MessageDrivenBean` and `MessageListener` interfaces. The `onMessage()` method of this example will receive a message sent to the queue "sonicQueue":

```
public void onMessage(Message msg) {
try {
System.out.println("Received from sonicQueue");
}
catch(JMSEException ex) {
ex.printStackTrace();
}
}
```

Pattern 2: Message-Driven Bean with a Topic Listener and Reconnect

This design pattern is used with the WLSAdapter to create and configure an MDB that will be used with a SonicMQ Topic to implement an XA transaction. It will support reconnect capabilities.

CONNECTION FACTORY

Use the WLSAdapter to create the XAConnectionFactory.

```
com.sonicsw.sonicmq.j2ee.wls.XATopicConnectionFactory
xafactory = new
com.sonicsw.sonicmq.j2ee.wls.XATopicConnectionFactory
("localhost:2506", "TopicMessageDriven",
"mypassword");
```

To identify the fail-over brokers, connection URLs for the factory are set:

```
xafactory.setConnectionURLs("localhost:2506,
localhost:2507");
```

A reconnect version of the connection factory is then created from the original connection factory:

```
long pingInterval = 1000L;
long facSleep = 10000L;
int maxIterations = 10;
```

```
com.sonicsw.ssps.reconnect.TopicConnectionFactory
rec_xafactory =
new
com.sonicsw.ssps.reconnect.TopicConnectionFactory
(xafactory, pingInterval, facSleep,
maxIterations);
```

Parameters are set to define reconnect behavior. The value of "pingInterval" indicates how often (in milliseconds) to check for a lost connection. The value of "facSleep" indicates how long to wait before reconnecting. Finally,

“maxIterations” indicates how many attempts should be made to reconnect. The reconnect version of the connection factory is then bound in the JNDI store:

```
m_context.rebind("myJMSrecxatcf", rec_xafactory);
```

DEPLOYMENT DESCRIPTORS

The values of the destination-jndi-name tag and the connection-factory-jndi-name tag reflect the topic and the connection factory, in this case “sonicTopic” and “myJMSxatcf”, respectively (see Listing 3).

Pattern 3: Stateless Session Message-Consumer Bean

This design pattern is used with the WLSAdapter to create and configure a stateless session bean that acts as a message consumer, an MCB. The example uses the receive() method of a receiver object to synchronously receive messages from a queue. The design would be similar for a topic-based MCB, where the receive() method of a subscriber object would be used.

This example assumes the home interface of the bean exposes three methods that correspond directly to the three types of receive() methods available on a receiver object:

```
public interface SonicMCBHome extends
javax.ejb.EJBObject
{
    public void receive()
        throws RemoteException, JMSEException ;
    public void receive(long timeout)
        throws RemoteException, JMSEException;
    public void receiveNoWait()
        throws RemoteException, JMSEException;
}
```

MESSAGE-CONSUMER BEAN

In the case of a queue-based MCB, the ejbCreate() method will look up a queue from which to create a session object (see Listing 4). From that session, a queue receiver object is created and the connection is started:

```
queue = (Queue) ctx.lookup(queueName);
receiver = session.createReceiver(queue);
connection.start();
```

The three receive() methods are implemented in the bean; each method corresponds to a matching receive() method signature on the queue receiver object (see Listing 5).

When one of these methods is called, the receiver will poll for a message from the queue. The receiver will wait indefinitely for a message if the first receive() method is called. If the second receive() method is called, the bean will wait an amount of time equal to the timeout parameter in milliseconds. If no wait time is desired, the

receiveNoWait() implementation is used. If the second receive() method times out, a null value is returned.

An MCB with a topic subscriber would be similar (see Listing 6). The receive() method is called from a topic subscriber instead of the queue receiver (see Listing 7).

Pattern 4: Message-Producing Bean

This design pattern is used with the WLSAdapter to create and configure a stateless session bean to act as a message producer. The MPB exposes one method, send(), in its home interface. The message is sent out on both a queue and a topic using an XA transaction to ensure that the send occurs on both destinations. The MPB also supports reconnect.

CONNECTION FACTORIES

First, XAConnectionFactories are created and bound. One is created for the queue and one for the topic, along with their reconnect counterparts (see Listing 8).

JNDI entries are created for the two destinations, “sonicQueue” and “sonicTopic” (see Listing 9).

DEPLOYMENT DESCRIPTORS

In the ejb-jar.xml file, the trans-attribute tag is set to “Required” for XA transactions (see Listing 10).

In the WebLogic-specific descriptor, however, the only necessary information is the ejb-name and the JNDI name for this stateless session bean. The connection factories and destinations aren’t referenced here:

```
<weblogic-ejb-jar>
  <weblogic-enterprise-bean>
    <ejb-name>JMSstatelessSession</ejb-name>
    <jndi-name>wls-jms-statelessSession</jndi-name>
  </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```

MESSAGE-PRODUCING BEAN

In the ejbCreate() method of the MPB, an initial context is created:

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
    "com.sun.jndi.fscontext.RefFSContextFactory");

env.put(Context.PROVIDER_URL,
    "file://localhost/C:/temp/jndi/fileStore/");

InitialContext ctx = new InitialContext(env);
```

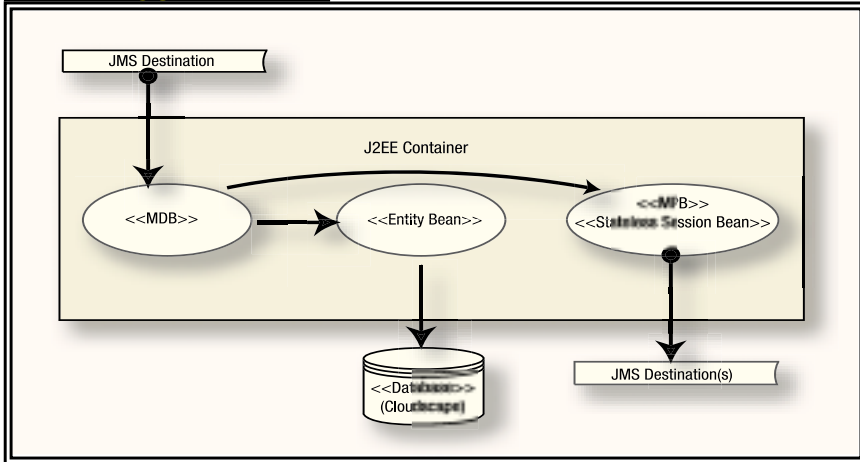
Then, a queue sender is created from the “myJMSxaqcf” connection factory, and a reference to the XAResource for the queue is retained (see Listing 11). The analogous steps are taken for a topic destination (see Listing 12).

At this point, a topic publisher and a queue sender are ready for use in the send() method.

Sun Microsystems

www.sun.com/servers/entry/beapromo3

FIGURE 1



MDB, entity bean, and MPB in three-bean interaction

When the MPB's send() method is invoked, a reference to the transaction object is obtained from WebLogic's Transaction helper:

```
javax.transaction.Transaction txn =
weblogic.transaction.TxHelper.getTransaction();
```

Next, the queue message is sent by enlisting the XA queue resource, creating and sending the message, and, finally, delisting the resource:

```
txn.enlistResource(qxar);
TextMessage qtext = qsession.createTextMessage
(qName + ":" + message);
sender.send(qtext);
txn.delistResource(qxar, XAResource.TMSUCCESS);
```

The steps are repeated for the topic:

```
txn.enlistResource(txar);
TextMessage ttext = qsession.createTextMessage
(tName + ":" + message);
publisher.publish(ttext);
txn.delistResource(txar, XAResource.TMSUCCESS);
```

A NOTE ABOUT SESSION POOLING

One clever aspect of the MPB built on a stateless session bean is the lightweight session pooling that can then be implemented through WLS. For a session bean, the initial and maximum

number of instances of the bean that reside on the WLS can be configured in the WebLogic-specific descriptor file for that bean (see Listing 13).

Although these parameters' values can't be changed while WLS is running, this interface allows for some control over resources and a simple pooling mechanism to optimize message production.

Pulling the Design Patterns Together

Figure 1 illustrates a circumstance where an MDB implementation consumes messages from a topic, processes them into a local database via an entity bean, and then produces a response message for transmission over a queue via an MPB. In a highly scaled environment, the transaction might also use pooled, auto-reconnect MPBs to achieve higher throughput and manage scalability. Most importantly, the entire interaction is transactional. If a failure occurs along any execution point, the entire process can be rolled back, with the developer coding minimal recovery logic.

This illustration is just one of many possible combinations of core JMS design patterns integrated with applications' EJBs. Dozens more are facilitated and documented in the adapter package. From this brief discussion of the design patterns and implementation techniques, it should be clear that along with a robust adapter suite such as that provided with SonicMQ, the tools exist to create reliable, high-performance enterprise solutions for most complex scenarios.

Conclusion

While BEA WebLogic Server has JMS support to address simple requirements, the need for sophisticated and higher-performance capabilities of pure-play JMS providers will remain in certain circles for quite some time. Fortunately, the leading JMS providers have worked around "WLS-isms" to enable transactional integration of their products.

In the meantime, BEA has stated that WLS 7.0 will solve the issues of standards compliance. This may indeed obviate the need for XA integration adapters. However, the issues of automatic reconnection, JNDI loading, and failure injection will remain, as will the need for advanced adapter solutions that address these issues. Moreover, having the design wherewithal, best practices, and proven design patterns to implement reliable, globally scalable messaging systems in enterprise application server networks, adapters or not, will be an enduring capability well beyond the retirement of WLS 7.0. The qualities of SonicMQ and its associated adapter for WLS stand out as unique among the field of solutions in this area.

Acknowledgement

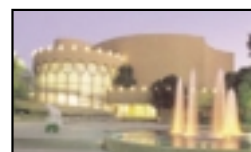
The authors wish to acknowledge the following individuals who contributed to this article: Gary Ark, Bill Cullen, Kathy Guo, May Hsu, K.V. Sastry, and Ademola Taiwo.

One quirk of WLS is that its JNDI repository isn't persistent across server sessions. Consequently, an outboard JNDI must be used or a JNDI loader class must be hard-wired for each application (the latter being the most common default). This seriously limits deployment flexibility and complicates network administration and maintenance. The Sonic WLSAdapter cleverly answers these issues by providing a pattern for a generic, XML-driven JNDI loader class and an accompanying utility that generates the XML file directly from the JMS broker management API.

Web Services Java XML NET Wireless



The Largest Web Services, Java, XML, .NET and Wireless Conference and Expo



CONFERENCE: OCTOBER 1-3, 2002
EXPO: OCTOBER 2-3, 2002
SAN JOSE McENERY CONVENTION CENTER, SAN JOSE, CA

Focus on Web Services

Those companies that get an early jump on Web services to integrate applications for the enterprise will be the winners in today's challenging landscape. Take three days to jump start your Web services education!

Focus on Java

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

Focus on XML

Here's your chance to hear up-to-the-minute developments in standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

Reaching Beyond the Enterprise

The Conference...

Organized into five comprehensive tracks: Web Services, Java, XML, Wireless, and IT Strategy
• Over 60 Information-packed Sessions, Case Studies and tutorials
• Thought leaders and innovators presenting the topics you most want to hear

Free...

FREE Web Services Tutorial to the First 100 to Register. Attend a FREE Web Services Tutorial on October 3. Register by August 15th to reserve your seat!

Who Should Attend...

- Developers, Programmers, Engineers
- i-Technology Professionals
- Senior Business Management
- C Level Executives
- Analysts, Consultants

The Largest Expo...

Over 75 exhibitors will deliver their best product demonstrations, stand ready to hear your challenges, and help you find solutions. Don't miss this important opportunity to expand your options.





A Conversation with Adam Bosworth



Adam Bosworth, vice president of engineering of the Frameworks Division at BEA, recently sat down with JP Morgenthal to talk about his role in WebLogic.



BY JP MORGENTHAL

AUTHOR BIO...

JP Morgenthal is coeditor-in-chief of *XML-Journal* and chief services architect at SoftwareAG. He has been writing and speaking about XML since 1997.

CONTACT...

jpm@sys-con.com

WLDJ: Tell us about your role at BEA.

Adam: Basically, I make sure that we build what's necessary for J2EE to become usable by the rest of us – on top of WebLogic Server. Whether it's building the user interface in the portal strategy, or building the overall development environment, WebLogic Workshop will make it easy for every reasonable developer to use.

WLDJ: What are your primary software development interests?

AB: I've spent my life trying to make building applications easy. Basically, helping developers build solutions – developing products, plumbing, or technology to help them build solutions. Whether it was Active Server Pages and the extent of my work at Microsoft, all of which was plumbing, or Access and working on the VB products, I helped them build the application. That's what I like doing.

WLDJ: What are the biggest challenges facing the developer community?

AB: We're moving into an era when more of the information on the Web will come to you, rather than you going out to it. Today we have a pull model. When you want to get information, you go to a site and pull the information out; information doesn't come to you. As we move into a world of mobile devices and applications interacting with each other, the pull model will become one of two models – push-and-pull architecture. A big challenge is in making it easy for developers to write and manage, and then administer and configure applications where the applications don't just provide information on demand, but send information to people.

Another challenge, even in building WebLogic Workshop, is making asynchronous Web services the linchpin of the product. Coming up with the right pro-

gramming model and making it easy was critical. We were able to work with mobile vendors and cut deals in weeks because they could see the opportunity for sending information to the devices. A big challenge is making the shift from a pull-model world to a push-pull model.

Also, we've built a large infrastructure for building complex Web-based apps, but most people don't have a good way to understand what the application is doing.

When you build a Web site you want to build four kinds of logic. There's logic that describes the overall site: what pages are brought up when, possible navigations, and how they depend on who you are and what you've done. That's site-level logic.

Then there's page-level logic. The user just filled these fields in. What should I do? How do I update my portfolio or my sales course automation?

Layout logic: What does the page look like? How conditional is that based on personalization and other things?

Finally, there's business logic. Right now, it's hard for developers to know where to put that logic, so they put it all in one place. You can't tell at any high level what's going on. I hear that it's happening in the .NET world as well. Making the business logic easy to separate and easy for business analysts who work with developers, so you can see the overall layout and flow of your logic and partition it in a natural and intuitive way, is the other major challenge we're facing. BEA's Framework Division is building products that provide solutions to these challenges.

WLDJ: How do you make partitioning easy?

AB: We're working with Java-Server Faces – working group JSR1-27. .NET has done it with controls. You can have UI code behind the page and that page code becomes strictly layout logic; the two can coexist. You can separate easily from your layout logic, which is just JSP in our world, and there will be code behind it. It's a baby step, but then you need an IDE to make it easy to use.

Another example is what we did in WebLogic Workshop with a two-part view. You can see what's going on in the Web service using the design view. Then you can go to the code. How do you handle incoming messages? The code is straight Java; you can toggle between the two views. Annotations make this possible. Metadata is becoming important because it can describe to the IDE what's going on and let the IDE provide higher-level views, like the design view in WebLogic Workshop. Another example is business process management.

WLDJ: Do you foresee the facility, like in C#, to store attributes as one of those areas?

AB: We've already done that in WebLogic Workshop. We use annotations and comments. JSR-175 has been submitted just for adding annotated metadata to Java. In addition, we submitted JSR-181, which is the JWS JRS itself and formalizes it for this particular case (Web services). The only difference is that in .NET it was done intrusively into the language. We do it more quietly, in comments. Endemically, in WebLogic Workshop we use metadata all over, in just that way, as attributes.

WLDJ: BEA has announced the BEA WebLogic Enterprise Platform and integrated stack based on WebLogic Server, which offers a single platform for Web services, portal, and integration capabilities. Why did BEA push to have a single application infrastructure?

AB: It's what developers need. It has scale. Once our customers put an outward face on an app, they need it to be available 24/7. You need high availability and complete reliability, hence the need for transactions. If you take those three needs – scalability, high availability, and transactions – you've got an app server. It's the kernel you need for pretty much any function I can imagine deploying today.

Portals, integration, and Web services are all part of the same thing. Customers have invested in IT over the last 10 years and built thousands of applications. We have customers that haven't been able to treat all these applications as intelligent resources they can integrate for both information and process. Now they're looking to build portals that act as UI integration layers that pull together the information in a personalized, customized way.

For their customers, employees, or partners, they're looking to pull together business processes so all the other application servers are either sources

of information or integral parts of the overall process. To do this, they're looking for the same kind of integration and high-availability scalability and transactions because they're mission-critical applications. If anything goes down or the system stops working, they stop delivering services to their customers, themselves, or their suppliers and partners.

This is the difference between how we look at integration and how other companies do, which is through business process management (BPM). I've seen a lot of people writing messages in integration brokers. They think of the message integration broker as just a switch that takes an incoming message and routes it to the right place. When we look at how our customers use BPM, they're integrating facilities within the app itself. They're calling EJBs at every step. The workflow is deeply integrated with how they choreograph and run the various components of the application. That's much easier to do with one integrated stack. It's hard to provide that kind of seamless integration from the outside.

That's an argument for a single platform. When you build one of these pieces everything is integrated, but it goes a little further. People are using Web services more to build the next generation of integration. I talk to companies making major bets on their EI system being replumbed on top of Web services over the next 12 to 24 months. They're asking themselves, "How do we wire the application logic to invoke or be invoked by these Web services? How do we work the UI logic to do the same?" They want it to be one seamless whole so they can also wire it into their user interface, but don't want to easily expose any functionality in the application. It's a fancy way of saying, "We need a single platform because our customers need it, because the applica-

tions they're building are about integrating information."

WLDJ: What part has BEA's acquisition of Crossgain played in providing BEA's unified application infrastructure?

AB: We've chosen to build something where the average corporate developer could focus on business logic and application logic, not plumbing. We saw an opportunity to increase use of J2EE if we could let people focus on the business application logic. Crossgain chose to start with Web services. Their role was to make Web services easy for developers in a way that fully exploited the J2EE platform. Our acquisition of Crossgain led to WebLogic Workshop. It's an environment that makes it easy to build truly enterprise-ready Web services. Crossgain and BEA had to do a lot of work to make sure that Web services lived up to what the enterprise would need.

WLDJ: Crossgain was based on a Web services platform? How viable is Web services technology, whether it's .NET or Java?

AB: Web services are extremely viable for one arena and becoming viable for another. They're extremely viable for integrating information within an enterprise. To do that, you need three things. You need coarse-grained messaging. You get that from Web services. You need a loosely coupled model because applications across the enterprise are run by different departments and groups and none of them can be redeployed in concert. You need to be sure that changing one application won't break the others. You need an asynchronous model because a great deal of what the applications do is such that you cannot assume that you can invoke them synchronously, either because they aren't available all the time or because there are peak loads that would overload your service.

You also need reliable mes-

saging, which isn't yet part of the Web services standard, but most enterprises are wired up to a message bus. There are bridges between message buses and we made JMS a transport for Web services. In doing so, we guaranteed that within the enterprise you could have reliable delivery, even within the context of Web services using JMS as the transport for the message buses rather than using HTTP. Now we give you a choice – use one, both, or either.

Security isn't as rich and well defined as it should be at this stage. Within the enterprise, security constraints were more manageable because you were safe within your firewalls. Some of the tools that are already there – right at your sign-in, un-sign-in using XKMS or SSL – are reasonable to do within the enterprise. If you look at where Web services are on B2B, that's where these adages are felt most

keenly. The two things that are slowing down Web services' adoption to the B2B space are the lack of standards for reliable messaging and an automatic assumption that all the stacks that implement Web services implement reliable messaging.

WLDJ: You said earlier that there's an assumption about the security model. What is it?

AB: Within the enterprise you tend to assume that someone who's made it within the firewall isn't dangerous. That doesn't mean they aren't dangerous; they aren't as dangerous as when you open things up through the firewall. You realize someone coming into this organization could be malignant because the code coming in isn't code written by your organization. That raises the ante dramatically. In one case you can do reasonable checks and balances and have speed bumps to catch people who are just being

sloppy on permissions and rights. The other case, when you really have to be completely confident that you have very hard-core security, is on the security across companies where people coming in simply aren't trusted in the same way because they don't work for you. That's where the demand goes up dramatically. We've been working with VeriSign and others to draft a new security directive that's come out of Microsoft, VeriSign, and IBM.

WLDJ: You also mentioned the importance of quality of service.

AB: We believe the best way to get quality of service is to support a model with queues. You can write intelligent tools to take things out of queues and process them in the order you need. If you simply spin up threads on machines every time a request comes in, you can't scale beyond the fixed limit of the hardware. A challenge for people on the Web has been

delivering a better quality of service to a high-profile customer than a low-profile customer. They're both coming in over the same Web gateway. At the end of the day, the Web service gateways are synchronous and spin up threads immediately in computers because they're synchronized. When you have a queue in the model, you can deliver very good quality-of-service characteristics. You can see how long something is sitting there. It's harder to see if it's spun up as a thread.

Reliable messaging is also part of quality of service. The customer has to decide. Reliable messaging usually comes at a price in terms of latency, in terms of speed – customers have to make the right trade-offs.

WLDJ: You worked for Microsoft and now you're with the Java camp. Do you see a difference in the approaches they take toward Web services?

AB: I worked for Microsoft for 11

years and played a role in getting Web services off the ground. The difference is in who the customer is. In the Java camp we think a great deal about enterprise computing and so the challenges we run into almost immediately are about enterprise computing. I'd say the Java community has taken off around enterprise computing and I think that's something that people miss a lot.

The Java community thinks a great deal about Web services as an integration platform and less about Web services as a consumer story. That's starting to change. Push Java is dominant on mobile devices because mobile devices are such a natural companion to Web services, they need a message-based model. Look at RIM as a success story for applications on mobile devices. We're starting to see a natural marriage and that's driving more of a consumer focus into how the Java community thinks.

Java has huge support among the customer base. Customers like the open model. Many bet on PowerBuilder years ago because it was the best for client/server computing. It had a lot of proprietary language. These days they're frustrated because there was only one vendor, which is not always the healthiest vendor. Our customers don't want that again, they want to know the language isn't controlled by any one vendor, and that there will be choices. That's Java's strength.

Microsoft's strength is that they control the language absolutely and proprietarily. They can move quickly to extend the language and to learn from Java, and they have. We'll see if Java in turn learns from what Microsoft has been doing and continues to extend itself. If it does, Java will be the dominant language for a long time to come.

We believe people can work together – the deep, hard-core systems programmers, the gurus, and the others who basically are writing business and application logic, which is what actually added value to their company.

Most of our customers don't beat their competitors by doing better plumbing. I see these two communities – systems programmers and applications developers – that want to work together. Having Java as the unifying story between them is powerful because it means each one can easily read and expand and augment what the other does in a very seamless way. I love watching one of my developers build a Web service using WebLogic Workshop, and then one of my other developers, a "VI and emacs, unless it's in the kernel I am not interested" guy, extends it to do something weird and wonderful that I'd never figure out. For us, Java is a lingua franca. The CLR model ultimately has the most value if multiple people are writing in multiple languages. Customers don't really want that. If all their developers are writing in different languages, then it's very hard to move code from one community to another.

Even if the CLR were open, and I'm a little skeptical, it's not clear that having a common language runtime per se is better.

What's interesting is how C# has learned from Java and innovated in its own way. If Java continues to extend and grow; if it learns how to natively talk to XML; if it learns how to natively add extensible metadata, and all the things that should happen; if it learns how to handle the fact that messaging is a fundamental core competency of language and that language should have constructs for rendezvousing messages, then Java will continue to be the dominant language in the enterprise, and probably on devices, because at the end of the day, mobile devices will dwarf PCs as well. On the other hand, if it doesn't, if it's declared to be perfect as is, and/or is height-bound by the fact that it's a community effort and takes more work to extend than one owned by a single very smart and aggressive vendor, then there could be risks.

Pick 3 or More and SAVE BIG!



Wireless Business & Technology • Java Developer's Journal
 .Net Developer's Journal • XML-Journal Web Services Journal
 ColdFusion Developer's Journal • WebLogic Developer's Journal
 PowerBuilder Developer's Journal

Get a **SPECIAL LOW PRICE** when you subscribe to 3 or more of our magazines

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTION



www.sys-con.com/suboffer.cfm

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany collaboration on a global scale

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

SPECIAL INTRODUCTORY OFFER
SAVE \$31*
 OFFER EXPIRES OCT 31, 2002

Now in More than 5,000 bookstores worldwide – Subscribe **NOW!**

Go Online & Subscribe Today!

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



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

You might not want to admit it, but have you ever lost or forgotten the password for the system user within WebLogic Server (WLS) 6.1? Or worse yet, accidentally deleted the fileRealm.properties or SerializedSystemIni.dat files?

Recovering from an Invalid System Password

AVOID PRODUCTION DISASTER WITH HIDDEN UTILITY

If so, you know the consequence – failure to produce a valid system password at server startup prevents the server instance from booting. This article shows you how to recover from this type of situation, keeping production downtime to a minimum.

Losing the System Password

As most administrators know, system password management within the production environment is critical. Root passwords are required to manage a wide array of operational resources, ranging from operating systems to network routers. A production environment is directly impacted when a key system password is lost, rendering the affected resource inaccessible until an administrator recovers or replaces the password.

With WLS, losing the system password within the default file security realm will prevent a server instance from starting up, blocking its boot process until the system password is reset or recovered. The problem is further complicated by the fact that WLS password management is promoted by the browser-based administration console, a valuable interface constrained by the limitation that the server must be up and running in order for it to be accessible. What options are available when the system password is lost but can't be reset from the console? The answer lies in a command-line feature hidden within the file realm itself.

Elements of the File Realm

Security realms play an important role within WLS 6.1; they define the set of users, passwords, groups, and access control lists used by a domain for its base security information. While various

types of security realms exist, WLS by default uses the file realm unless configured otherwise.

For a given domain utilizing the file realm, two crucial files work together and constitute the underlying repository of the security realm. The first, a text file, named fileRealm.properties, persistently defines the actual set of users, hashed passwords, groups, and access control lists of the realm. The second, a binary file named SerializedSystemIni.dat, defines the seed input used by the file realm when a cleartext password is hashed in its SHA-based format. Both files exist within the root directory of the given domain (e.g., ./config/mydomain).

Since the SerializedSystemIni.dat file encapsulates a time-variant encrypted key created when a domain is first generated, no two SerializedSystemIni.dat files are created alike for a specific server installation. Therefore, when a cleartext password is hashed for a certain user and stored within the fileRealm.properties file, an explicit relationship is immediately established between the fileRealm.properties repository and its input SerializedSystemIni.dat file. The password stored in the repository and hashed with the SerializedSystemIni.dat input can be successfully compared only to its cleartext equivalent hashed with the same SerializedSystemIni.dat. This element is critical to the validation process of password comparison, where a cleartext password is compared to a stored password at the hash level, since it's impossible to directly reverse the hashed password back out to its original cleartext format.

An Invalid Password – More Than Just Forgetful

With WLS 6.1, the system password can be invalidated in a few ways. Aside from the obvious cases where the password is forgotten or misplaced (considered a lost password), the system password for the file realm of a given domain also becomes invalid when one of the following situations occurs:

- **Errant password modification:** The fileRealm.properties file has been modified in such a way that an errant change is introduced to the hashed password value for the system user (intentional or otherwise).
- **Mismatched seed and repository:** The SerializedSystemIni.dat file has been modified, replaced, or removed entirely, introducing a change that breaks the relationship between a given salt file and the set of passwords stored within a specific fileRealm.properties file.

An errant modification to a password is equivalent to password corruption. It usually results

from one of two possibilities, differing only by the intent – an accidental change by an administrator who inadvertently modifies a password during an edit session, or a planned attack by a malicious user who intentionally corrupts a password. In both scenarios, the original hashed value of the password is changed at the character level so that a new hash is created, making it impossible to determine what the original cleartext of the new hash is and rendering the password invalid.

In the scenario involving a mismatched seed and repository, the relationship between a given salt file and the set of passwords stored within a specific fileRealm.properties file is broken as a result of the SerializedSystemIni.dat seed file having been modified, removed, or replaced entirely by another file (seed or otherwise). When the seed file becomes invalid with respect to its relationship in this manner, password comparison at the hash level will fail, since passwords stored within a given fileRealm.properties file and initially hashed with one salt are now compared to passwords hashed with another salt.

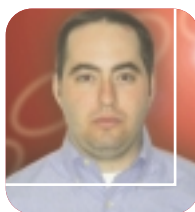
Available Recovery Options

When the system password is invalidated by one of the above possibilities, the recovery options are limited. If the password has been lost or corrupted,

an administrator can restore the password from a backup copy of the fileRealm.properties file, if one exists. If a seed file is invalidated and its relationship to the set of passwords within a given repository has been broken, an administrator can restore the salt from a backup copy of the SerializedSystemIni.dat file – again, if the backup exists.

What can be done, however, when backup copies of the seed and repository files don't exist for the affected domain? You can copy the hashed passwords and appropriate files from the file realm of another domain (where the values of those passwords are known) when another domain exists, or create a new domain by installing a clean instance of WLS and copying from that. Of course, this has problems in its own right, as it depends upon the existence of another domain or the creation of a new domain through installation, and involves a fair amount of manual interaction. Besides, you might not want to risk compromising a production environment with passwords from another configuration.

When backup restoration isn't available and domain copies are not possible or desired, the best recovery option is to reset the password from the command line using the features of a hidden utility within the file realm itself. Mastering this utility will ensure proper recovery from disaster and enable a



BY STEVE MUELLER

AUTHOR BIO...

Steve Mueller is a principal consultant for BEA Systems, where he specializes in the design, development, and administration of enterprise systems running on WebLogic Server.

CONTACT...

smueller@bea.com

CUT OUT AND COMPLETE FORM ON BACK

FREE

EXPO PASS!

\$75 VALUE

web services
conference & expo

JDJ
conference & expo

XML
conference & expo

wireless
conference & expo

OCTOBER 1-3, 2002
SAN JOSE McENERY CONVENTION CENTER,
SAN JOSE, CA

Look Under the Hood and Beyond the Hype...

DIVE INTO WHAT'S HEADED YOUR WAY IN JAVA, XML, .NET, WEB SERVICES AND WIRELESS

Who Should Attend:

- Developers
- Programmers
- Architects
- Engineers
- C-Level Executives
- i-Technology Professionals

EXPO HOURS:

OCT. 1.....11:00 AM – 6:00 PM

OCT. 2.....11:00 AM – 6:00 PM

OCT. 3.....11:00 AM – 4:00 PM

Conference & Expo Features

- The Most Significant Web Services and Wireless Technology Event of the Year!
- Sun Java™ Fast Path and Sun Java University™ Classes
- Two Concurrent Conference Programs for Just One Registration Fee
- The Largest Web Services, Java, XML, .NET and Wireless Expo in the World!

Platinum Sponsor:

Silver Sponsor:

Educational Sponsor:

Gold Sponsors:

Corporate Sponsor:

REGISTER ONLINE: WWW.SYS-CON.COM WITH CODE 8102

production environment to operate with little interruption. The rest of this article discusses in detail this utility – the FileRealm class.

The FileRealm Class

Every type of security realm within WLS 6.1 is ultimately managed and represented by a pure Java class. Such classes are either provided with the product out of the box or are externally developed to meet specific security needs. The default file realm is no exception, and is governed by the internal `weblogic.security.acl.internal.FileRealm` class that ships with WLS (packaged as part of the core `weblogic.jar` file). The primary responsibilities of this class are to manage the users, passwords, groups, and access control lists of the file-based security realm and provide authentication and authorization services for clients, using the underlying `fileRealm.properties` file as its persistent security repository and master record of data.

By all accounts, the FileRealm class should offer nothing more than a set of interface methods that other internal components of WLS can publicly use when authenticating and authorizing client access within the security realm – and it does. Interestingly enough, however, the FileRealm class also exposes a main method, allowing it to be invoked from the command line as a normal exe-

cutable program. As a result, the internal class is surprisingly dual-purposed, serving primarily as a supporting library class to other elements within WLS, yet unexpectedly promoting itself as a command-line utility at the same time.

When executed from the command line, the FileRealm class provides a mechanism by which a cleartext password for a given user can be hashed with a specified input seed file, producing an encrypted equivalent output that can be stored within the `fileRealm.properties` repository and subsequently referenced by the file realm as the actual password for the given user. In this way, the class effectively gives administrators the ability to set passwords from the command line for any user within the file realm, free of the limitations imposed by the browser-based administration console and the password management system it offers. The potential value offered by this internal class is only fully realized in the most disastrous of situations, where the system password has been lost or invalidated and must be reset from the command line when the server consequently fails to start.

Resetting the System Password

Utilizing the command-line feature of the FileRealm class to set or change a password for a user within the file-based security realm is rela-

tively straightforward. You must first identify the set of users that require password changes – the FileRealm class allows you to consolidate all password changes for multiple users at once, so executing the program in multiple iterations to handle a batch of users on an individual basis isn't required. However, in the example below we'll change the password for only one user (the system user).

When the set of users has been identified, a properties file must be created, and the users need to be defined within this file. Comments are allowed within the file (preceded by the # sign), and for each user within the set, a corresponding entry must be defined on its own new line within the file using the following format:

```
user.<username>=<cleartext_password>
```

An example follows:

```
# define the user system to have the cleartext password weblogic
user.system=weblogic
```

This file provides the input list of users for the FileRealm class to process and associate each user with its original cleartext password. For this reason, the properties file can also be referred to as the input definition file. This file can be located anywhere in the local file system, and must end with the `.src` extension. In the example used below, we'll define the input definition file as `user.properties.src`.

When the input definition file has been created, the location of `SerializedSystemIni.dat` must be determined before the FileRealm class is executed from the command line. Recall that `SerializedSystemIni.dat` provides an input seed (or salt) to the hashing phase of the encryption process, and that a password hashed with a specific salt can be successfully compared only to another password hashed by the same exact salt. Therefore, the `SerializedSystemIni.dat` file, localized at the domain level and present within the `config/<domain>` directory, has an explicit relationship to the passwords it hashes for the file realm of a given domain. Since WLS can't maintain a file realm in which the set of hashed passwords present have been seeded by different `SerializedSystemIni.dat` files, you should never mix passwords hashed by different salt files within the same file realm.

Depending upon the situation and the specifics of your problem, the FileRealm class can use an existing `SerializedSystemIni.dat` file for its input seed or create a new one if none exists. Referencing an existing `SerializedSystemIni.dat` is beneficial in the scenario in which the system password for a given domain has been lost but the domain-level salt file is still present and valid. This forces the FileRealm class to create a new password with the same seed used to hash all other passwords within the domain. Creating a new `SerializedSystemIni.dat` is useful when the salt file has been deleted or corrupted and the relationship to the passwords hashed by the original salt has been broken, making them invalid. Note that when a new `SerializedSystemIni.dat` file is created by the FileRealm class and used to hash new passwords for a domain, all other passwords within that same domain must be reset as well, since the seed for the hash has changed.

Executing the FileRealm class is simple and straightforward. Set your system classpath to include the `weblogic.jar` file from the lib directory of the WLS installation and invoke the utility from the command line as follows:

```
java weblogic.security.acl.internal.FileRealm \
<path_to_output_file> \
<path_to_salt_file>
```

Upon processing the input definition file, the FileRealm class generates its resultant set of users and their hashed passwords into the output file defined by the first parameter. This file is identical to the input definition file in every way except for the format of the passwords – the passwords in the output file are hashed, while those from the input are cleartext. Of course, this is the desired result. The path to the output file given by the first parameter should be identical to that of the input definition file, with the `.src` extension dropped for the output file. For example, if you had created the input definition file from above at `/tmp/user.properties.src`, you'd now need to define the output file at `/tmp/user.properties`. This accommodates the way the FileRealm class internally handles the location of the input definition file – it concludes its location by using the same path and name of the output file, yet it assumes the input file has the `.src` extension added to it.

The second parameter defines the location of the `SerializedSystemIni.dat` file. If the relative or absolute path specified references an existing salt file, then that seed is used as the input. If not, a new seed is created at the path specified with the given filename. Note that while it's possible to reference or create a new salt file with a name other than `SerializedSystemIni.dat` and outside of a given domain directory, WLS will recognize at runtime only seeds named `SerializedSystemIni.dat` and placed within their appropriate domain directories.

Upon execution, the FileRealm class will generate its hashed password outputs into the file defined by the first input parameter. When the command-line utility has completed, the invalid passwords found within the `fileRealm.properties` repository of the affected file realm should be overwritten with the new passwords stored within the output file. This last step should be done manually using your favorite text editor and a few select copy/paste operations. Once the changes have been made to `fileRealm.properties` and the file has been saved, your invalid passwords are valid once again, and must be verified. Start the server and provide the password recently set for the system user. If the password recovery was successful, the server will properly start and WLS will operate as expected. When you're finished, remove the input definition file and its output equivalent from the filesystem to avoid compromising your newly set passwords.

Conclusion

The browser-based administration console, an excellent utility constrained by the requirement of a running server, is the supported mechanism for password management within WLS 6.1. When the system password has been lost or invalidated and WLS won't start, however, the console becomes inaccessible and the only reliable option for resetting a file realm password is to do so from the command line. Utilizing the hidden features of the FileRealm class, you can manage passwords from the command line and recover from an otherwise disastrous situation. Additionally, while the hidden features of the FileRealm class discussed within this article relate to WLS 6.1, they are fully accessible and available under WLS 7.0, and should be used for the same purposes when running in the 6.1-compatible security mode.

TO REGISTER:

ONLINE
WWW.SYS-CON.COM

FAX
201-782-9651

MAIL
SYS-CON EVENTS, INC.
135 CHESTNUT RIDGE ROAD
MONTVALE, NJ 07645

CALL
201-802-3069

web services EDGE conference & expo
JDI EDGE conference & expo
XML EDGE conference & expo
wireless EDGE conference & expo

OCTOBER 1-3, 2002
SAN JOSE McENERY CONVENTION CENTER,
SAN JOSE, CA

YOUR INFORMATION (Please Print) Mr. Ms.

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

A. Your Job Title
 CTO, CIO, VP, Chief Architect
 Software Development Director/Manager/Evangelist
 IT Director/Manager
 Project Manager/Project Leader/Group Leader
 Software Architect/Systems Analyst
 Application Programmer/Evangelist
 Database Administrator/Programmer
 Software Developer/Systems Integrator/Consultant
 Web Programmers
 CEO/COO/President/Chairman/Owner/Partner
 VP/Director/Manager Marketing, Sales
 VP/Director/Manager of Product Development
 General Division Manager/Department Manager
 Other (please specify) _____

B. Business/Industry
 Computer Software
 Computer Hardware and Electronics
 Computer Networking & Telecommunications
 Internet/Web/E-commerce
 Consulting & Systems Integrator
 Financial Services
 Manufacturing
 Wholesale/Retail/Distribution
 Transportation
 Travel/Hospitality
 Government/Military/Aerospace
 Health Care/Medical
 Insurance/Legal
 Education
 Utilities
 Architecture/Construction/Real Estate
 Agriculture
 Nonprofit/Religious
 Other (please specify) _____

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

Location	Company
10,000 or more	01 <input type="checkbox"/> 01 <input type="checkbox"/>
5,000 - 9,999	02 <input type="checkbox"/> 02 <input type="checkbox"/>
1,000 - 4,999	03 <input type="checkbox"/> 03 <input type="checkbox"/>
500 - 999	04 <input type="checkbox"/> 04 <input type="checkbox"/>
100-499	05 <input type="checkbox"/> 05 <input type="checkbox"/>
99 or less	06 <input type="checkbox"/> 06 <input type="checkbox"/>

D. Please indicate the value of computer and communications products and services that you recommend, buy, specify or approve over the course of one year:

<input type="checkbox"/> \$10 million or more	<input type="checkbox"/> \$1 million - \$9.9 million
<input type="checkbox"/> \$500,000 - \$999,999	<input type="checkbox"/> \$100,000 - \$499,999
<input type="checkbox"/> \$10,000 - \$99,999	<input type="checkbox"/> Less than \$10,000
<input type="checkbox"/> Don't know	

E. What is your company's gross annual revenue?

<input type="checkbox"/> \$10 billion or more	<input type="checkbox"/> \$1 billion - \$9.9 billion
<input type="checkbox"/> \$100 million - \$999 million	<input type="checkbox"/> \$10 million - \$99.9 million
<input type="checkbox"/> \$1 million - \$9.9 million	<input type="checkbox"/> Less than \$1 million
<input type="checkbox"/> Don't know	

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

G. Which of the following products, services, and/or technologies do you currently approve, specify or recommend the purchase of?

<input type="checkbox"/> Application Servers	<input type="checkbox"/> Web Servers
<input type="checkbox"/> Server Side Hardware	<input type="checkbox"/> Client Side Hardware
<input type="checkbox"/> Wireless Device Hardware	<input type="checkbox"/> Databases
<input type="checkbox"/> Java IDEs	<input type="checkbox"/> Class Libraries
<input type="checkbox"/> Software Testing Tools	<input type="checkbox"/> Web Testing Tools
<input type="checkbox"/> Modeling Tools	<input type="checkbox"/> Team Development Tools
<input type="checkbox"/> Installation Tools	<input type="checkbox"/> Frameworks
<input type="checkbox"/> Database Access Tools/JDBC Devices	<input type="checkbox"/> Application Integration Tools
<input type="checkbox"/> Enterprise Development Tool Suites	<input type="checkbox"/> Messaging Tools
<input type="checkbox"/> Reporting Tools	<input type="checkbox"/> Debugging Tools
<input type="checkbox"/> Virtual Machines	<input type="checkbox"/> Wireless Development Tools
<input type="checkbox"/> XML Tools	<input type="checkbox"/> Web Services Development Toolkits
<input type="checkbox"/> Professional Training Services	<input type="checkbox"/> Other (Please Specify) _____

\$75 Value
 Complete and return this form to receive FREE admission to the Expo and all FREE Show Features.
 After September 20, 2002, bring this pass on site for FREE admission.
 PLEASE PRINT CLEARLY (PHOTOCOPY FOR ADDITIONAL REGISTRANTS).

JBuilder 7 Enterprise

BY BORLAND SOFTWARE CORPORATION

Borland®

Reviewed by Andy Winskill

Borland Software Corporation

100 Enterprise Way
Scotts Valley,
CA 95066-3249

Phone: 831 431-1000

Web: www.borland.com

E-mail: customer_service@borland.com

Pricing

New User \$2,999,
Upgrade \$1,899

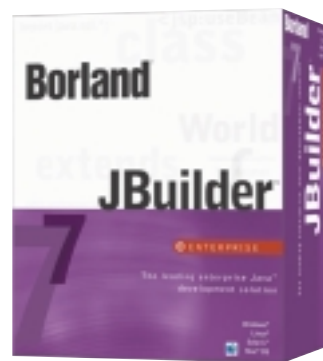
Test Environment

Sony VAIO PCG-GRX316MP, 512MB RAM, Windows XP Professional

Given the current pace of technology change, we seem to be under ever-increasing pressure from product vendors to upgrade to the newest, latest version of software. Most software vendors now seem to have major software releases scheduled every six months. This can put a strain not only on you, but also on your finances. Along with JBuilder 7, Borland has introduced another companion product, Software Assurance – but more on that later.

I come across two types of developers in my consultancy engagements – those who like to use an IDE, and those who frown on the technology, preferring to use the likes of Notepad (but more likely Emacs!) to produce the Java source, and then some custom build process (often using Ant) – what I describe as a “roll-your-own” development environment – to produce the application. I can see benefits to both approaches, and I’ve often thought that either approach to development environments is a

matter of personal or corporate preference. I do, however, believe that a good IDE can be a great aid to a development team’s productivity. A good IDE is one that has sufficient flexibility to enable the development team to work together and build the software solution. The features of the IDE should never constrain the solution the development team wishes to create!



JBuilder was one of the first IDEs for Java and is now probably the most prevalent. However, things are becoming much more competitive in the IDE marketplace, with the likes

of the Eclipse and NetBeans initiatives and low-end IDEs such as IntelliJ IDEA from JetBrains (www.intellij.com). Having been a JBuilder user for a number of years, I’d been investigating switching IDEs, so I was very interested in seeing what JBuilder 7 would bring.

Installation

The JBuilder distribution comes in a rather impressive box. A number of paper-based manuals describe JBuilder’s features and walk you through the various wizards that form JBuilder. These manuals are high-quality and very easy to follow. Unpacking the CDs, I found a number of products supplied as part of JBuilder Enterprise: JBuilder itself, a CD of companion tools (a set of components and plug-ins provided by Borland partners), a development version of Borland’s Enterprise Server, and a trial version of the Optimizeit Suite. A rather impressive set of CDs!

Installing JBuilder was a breeze; I opted for a complete installation and an installation of the Borland Enterprise Server (see Figure 1). The install program worked fine in my environment (I’d had a couple of issues with JBuilder 6) and, with a couple of CD swaps, I had JBuilder and BES installed. The complete JBuilder installation is rather large, taking more than 850MB of disk space. This does, however, account for BES and the Optimizeit Suite; JBuilder itself takes more than 420MB of space.

JBuilder Overview

One of the major benefits to using JBuilder is that it comes with a number of project wizards. After creating a new project you can choose to create a large number of different artifacts from the “Object Gallery.” The Object Gallery provides a simple interface to allow you to choose which wizard to run. Of interest to WLS developers is

Borland

<http://info.borland.com/new/javasolutions/92115.html>

the Enterprise tab, from which you can choose to access a number of EJB features. Web applications, servlets, and JSPs can be created via the Web tab on the Object Gallery and JUnit tests can be created via the Test tab. To show some of these features, I'll walk through the creation of a simple EJB component.

Assuming the destination platform is WLS 7.0, I'll create an EJB using the EJB 2.0 specification. To create such an EJB in JBuilder, select the EJB 2.0 Bean Designer, which Borland lists as a key feature of JBuilder Enterprise. After the wizard has created an EJB module for my components (a build artifact that includes the EJB JAR and the deployment options) I can create the actual EJB components. This is done in the visual EJB designer, which gives the option of creating each of the current EJBs in the spec.

Using the visual designer to create an EJB is tremendously easy. The designer allows the developer to configure each and every option. This has a direct effect on the EJB code, as well as the deployment descriptor. In Figure 2 we can see that JBuilder can create session EJBs as well as entity EJBs. The power of the visual designer becomes very apparent when

it's used to create relationships between entities. This tool gives developers a clear vision of what they are creating, as well as giving full access to the properties of the entities and the relationships between them. Excellent!

When the EJB frameworks have been created we can edit the source code. In JBuilder 6, refactoring features were introduced that allowed the simple renaming of classes, operations, and attributes. In JBuilder 7 these features have been extended to include the extraction of methods from blocks of code and surrounding blocks of code with the appropriate try/catch blocks. These features work really well and are a valuable aid to developer productivity, especially if the development team is following an extreme programming-style development process.

When the code compiles and the components are ready to be deployed, deployment can be done from within the IDE. This feature makes use of the WLS command-line deployment tool, weblogic.Deployer. Within JBuilder you can configure the options for this feature, allowing, for example, the deployment of the application to a cluster or to different servers. However, before deploying the application the server needs to be configured from within the IDE. Figure 3 shows the options available to the developer for a WLS 7 installation. JBuilder supports WebLogic Server 5.1, 6.x, and 7.0.

JBuilder 7 includes a UML code visualization feature that provides a UML view of the class currently being edited and its associations. This feature is useful if the developer has a model to compare to, but it's not a UML design tool like Rational Rose/XDE or TogetherSoft's TogetherJ. I don't find this to be a problem, and I applaud Borland for introducing design features into their product, as I often come across developers who

wouldn't know one end of an association from another.

New Features

As elements of a Java-based architecture evolve and new versions of the JDK and other APIs are released, a roll-your-own development environment can be quick to react. Having full control of your development environment you can plug in the latest version of the JDK and carry on development. With JBuilder 7, the build environment is opened up, and you can add your own custom build tasks, and even build your project using Ant. The benefits of this are tremendous, as you can now make use of third-party build processes. For example, you can now use JBuilder to create Web services for WebLogic Server 7 using the provided custom Ant tasks.

Within the JBuilder environment you can specify at which phase of the build process the Ant tasks apply, or you can run the Ant tasks manually, bypassing the JBuilder build process. This is really great flexibility if it's desired; if it isn't, you can just use the JBuilder build process.

The addition of Ant integration into the IDE build process is a feature that sets JBuilder apart from most current IDEs. Borland lists JBuilder's key features as:

- A visual EJB designer
- Deployment to leading application servers, including WebLogic Server
- Support for Web services development
- Web application development and deployment with JSP and servlets
- UML code visualization
- Refactoring and unit testing
- Configuration management

The complete list of features can be found on the Borland Web site, but I'll mention a couple that I've found very useful. As I mentioned before, there's the integration with Ant, plus there's now support for SQL/J, as well as support for WebLogic Server 7.

Unfortunately, the WLS 7 support is added as an option to the WLS 6 server properties. This means you can't easily switch between application server versions. This is a pain when you still have clients working on both 6.1 and 7.0 as you have to reconfigure your server properties. A completely separate server configuration for WLS 7 would have been much nicer.

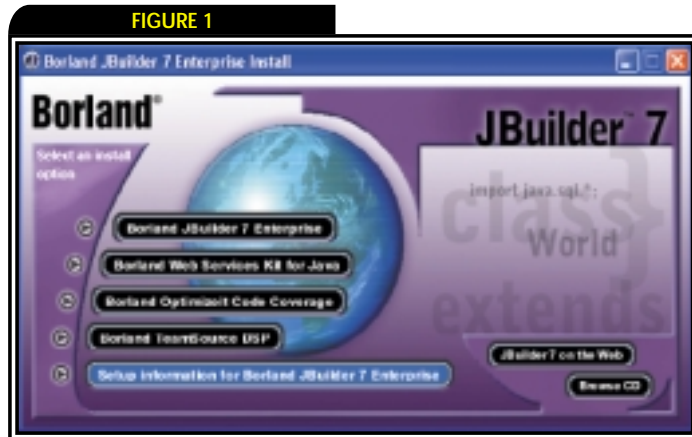
Numerous additional small features that make the developer's life easier have been introduced in JBuilder 7. For example, you can now close a code window from a button on the Window tab rather than having to right click and select close; it's a small thing, but I always found the old way irritating.

With JBuilder 7 Borland introduced Software Assurance. There were only 12 months between the release of JBuilder 5 and that of JBuilder 7. Previously, the only option was to purchase the upgrade product for a price slightly below that of the full version (in the UK), as software maintenance wasn't available. This business model may have been okay when products were on a 12-18 month upgrade program, but when new major versions of the product are released every 6 months, it becomes a strain. For a development team these costs can quickly become a considerable proportion of the development costs. Software Assurance addresses these concerns. Software Assurance is a comprehensive software maintenance product that gives us peace of mind concerning the JBuilder product set for the next couple of years. It's great to realize that Borland's been listening to its customers!

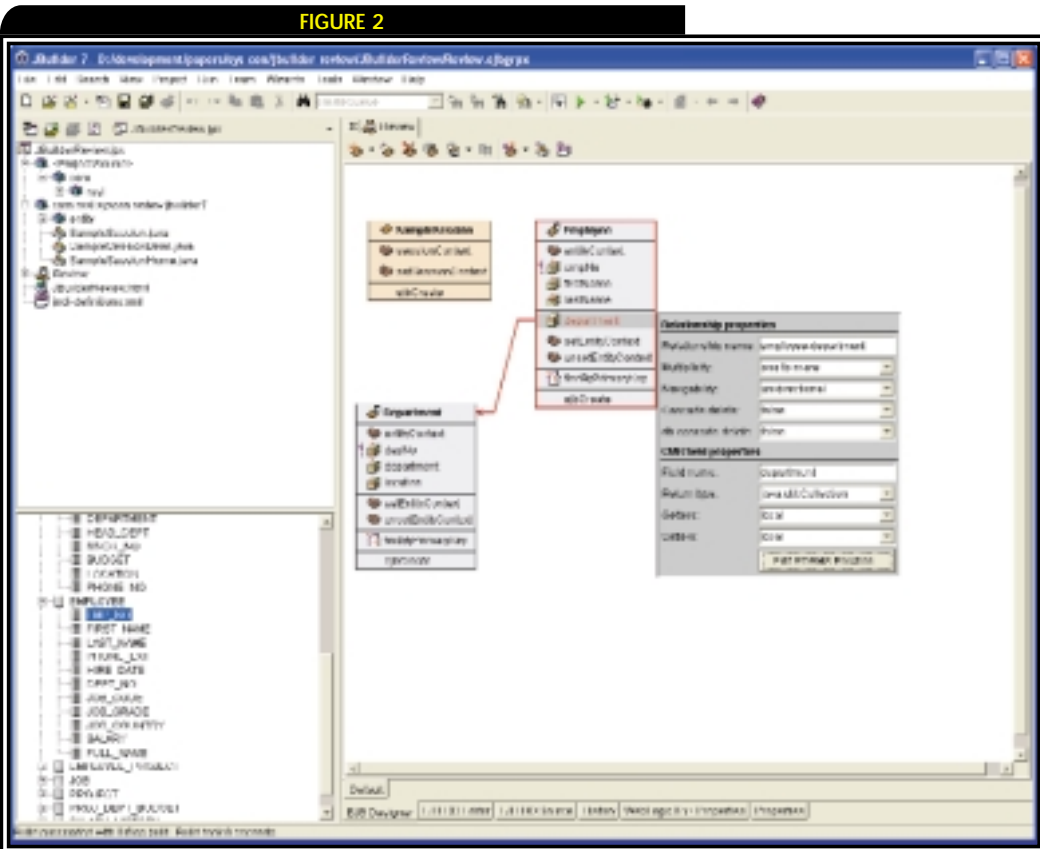
Summary

With the release of JBuilder 7 Enterprise, JBuilder has become a truly excellent product. JBuilder 6 was a hard act to follow, but JBuilder 7 accomplishes all it sets out to achieve and

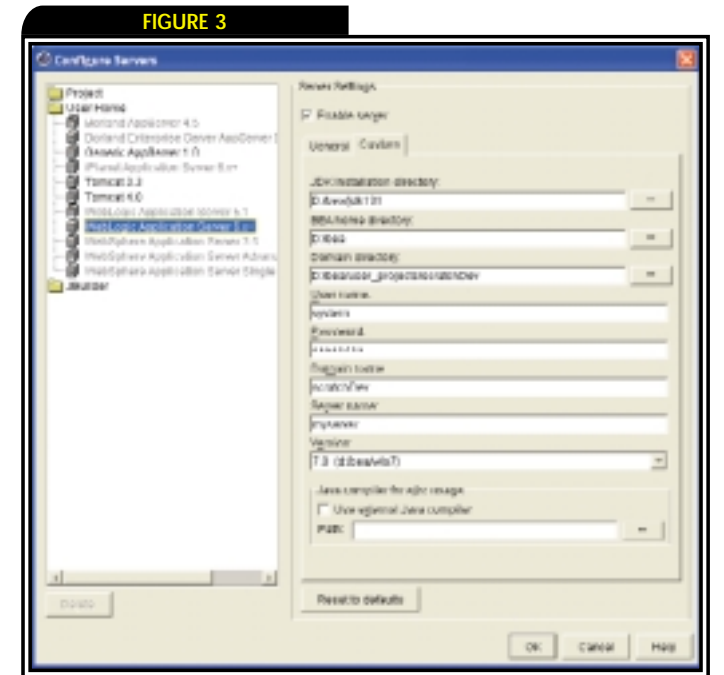
more. It introduces many new features and supports JDK 1.4. Many tools have tried to market themselves on developer productivity; however, I believe that JBuilder 7 Enterprise will help any J2EE development



The JBuilder Install



Creating a collection of EJBs



Configuring WLS options in JBuilder

team become much more productive. With this release Borland has opened up the JBuilder build process, providing the development team with a very flexible environment for introducing new technology. This should keep most developers happy! 🍌

AUTHOR BIO

Andy Winskill is a principal consultant at Rosewood Software Services Ltd., UK. He specializes in BEA and Rational software and has more than 10 years of experience designing and constructing EAI and B2B applications. Rosewood Software Services is a BEA partner and a Rational partner.

CONTACT...

andy.winskill@rosewoodsoftware.com

WebLogic on the Mainframe

JUMP-START YOUR WEBLOGIC DEPLOYMENT



BY

TAD STEPHENS & ERIC GUDGION

AUTHOR BIOS...

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

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

CONTACT...

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

In helping our customers deploy J2EE applications on the mainframe we've learned a number of tips and tricks. We've included configuration settings, tuning suggestions, and descriptions of existing production applications in this article. Although each environment is different, these tips and tricks should jump-start anyone considering a mainframe WebLogic deployment.

In the first article (*WLDJ*, Vol. 1, issue 7) in this series, we discussed many of the business benefits to be realized by deploying J2EE applications on the mainframe. These benefits included leveraging Java for better programmer productivity, aggregating multiple servers onto a single mainframe partition to lower operational costs and more efficiently utilize existing hardware, leveraging mainframe quality-of-service capabilities for 24x7x365 application availability, and extending existing applications and data located on the host machines. The second article (*WLDJ*, Vol. 1, issue 8) detailed how to install and configure WebLogic Server for z/Linux and z/OS environments, including the steps required, the resources needed on the mainframe, and the differences from installing WebLogic on other platforms.

One of the benefits that can be realized when deploying WebLogic Server on the mainframe is the extension of access to existing systems and data. In today's business environment enterprises are looking more than ever for ways to leverage existing investment in mainframe systems and databases rather than taking on the costs associated with rewriting applications and rehosting them in a distributed environment. Web services is a key technology that can enable this access. Rather than covering Web services and data integration in this article, we've decided to add a fourth article to our trilogy, à la Douglas Adams and *The Hitchhiker's Guide to the Galaxy*, to thoroughly detail how to Web service-enable existing mainframe applications and data using WebLogic Server.

Now let's get to it.

Performance Tips

When it comes to tuning applications, no recommendation will fit all customers. In general, a baseline for an application should be created, including a well-defined test procedure that exactly or closely models the behavior of the business application. All tuning and application changes can then be compared to the baseline by rerunning a well-defined test procedure. Once tested and validated, changes that result in performance improvements can then be promoted to the production system with risks minimized. This performance and tuning methodology requires establishing a test environment in which the configuration can be controlled, along with defining and implementing a repeatable test process.

However, a few generalizations can be made about performance and tuning for WebLogic Server running on the mainframe. These tips will help create a good starting point for creating a baseline.

Hardware Requirements

There are a number of factors that affect the performance of a WebLogic-based application on the mainframe. Some of these affect the operating system, some the application and security subsystems, and some are related to WebLogic Server. However, none are more important than whether the underlying processor is designed to support Java. Specifically, IBM recommends the G5 class processor with IEEE floating-point support for Java applications to achieve optimal performance. Although Java applications can execute and be deployed on a non-IEEE floating-point processor, the performance is significantly lower. One alternative is to use non-IEEE floating-point processors for development or prototype work where overall throughput is not a critical factor and G5 class processors for production deployment.

GENERAL TUNING

We've broken the tuning topic into two sections, z/Linux and z/OS, based on the operating system used. Although we make some generalizations, these suggestions are an excellent starting point for planning a mainframe WebLogic deployment. In addition, there are some UNIX System Services (USS) parameters that should be reviewed. The actual changes made to your environment will depend on a number of factors, such as the workload you will be processing and other applications deployed. In particular, the workload – concurrent users, number of transactions, and time period –

greatly affects the decisions you have to make when configuring a system.

Tuning Tips for z/Linux

- Set the virtual machine guest size to 512MB. This is a good average size for initial configuration, although you might be able to create a smaller machine if your workload and concurrent user load are relatively small.
- Disable any Linux services that aren't needed.
- As the virtual machine hosting WebLogic Server will have quite a few interactive users, we recommend that an execution class be assigned to WebLogic, ensuring that the server will have enough CPU and memory resources.
- Ensure that the WebLogic NativeIO option is enabled. This can be set from the WebLogic Console using the Tuning tab.

Tuning Tips for z/OS

- Consider placing commonly used modules (javac for example) into the LPA (link pack area).
- Follow the TCP/IP tuning recommendations for your operating system release.
- Use the WorkLoad Manager to ensure that the right mix of system resources is used.

USS PARAMETERS

WebLogic executes as a USS task. In fact, it is not uncommon to find that WebLogic is the first major application to execute in this environment. Because of this, the USS configuration should be reviewed prior to deployment and adjusted as needed. The BPXINITxx member in the SYSx.PARMLIB library contains the parameters that control the execution of USS tasks.

As a baseline, the following parameters should be reviewed:

- **MAXASSIZE:** This is the maximum address space size. If resources will allow, set this to the 2GB maximum (2147483647).
- **MAXTHREADS:** This is the maximum number of threads per process. A good starting point is 10000.
- **MAXTHREADTASKS:** This is the maximum number of operating system tasks a given address space can have active concurrently. Setting this parameter to 5000 is a good starting point for this value.

SECURITY CONSIDERATIONS

In addition to the settings noted above, the user identity used to start WebLogic Server can also affect the application's performance. A number of parameters that affect USS resource allocation are set in the user's RACF (Resource Access Control Facility) profile. These values may override those defined globally for USS, impacting WebLogic Server performance. There are a num-

ber of ways to prevent this from happening, such as removing the RACF USS parameters or setting the global USS parameters lower and configuring higher values in the user's RACF USS profile. However, the final implementation is administrator's choice.

The first parameter to check for the user identity starting WebLogic Server is the personal address space size value ASSIZEMAX, specified in the RACF USS segment. This parameter sets a specific user's address space. If this value is less than the MAXASSIZE, then the smaller ASSIZEMAX value associated with the user's profile will override the MAXASSIZE and be used instead. As a workaround, many administrators will set the global MAXASSIZE parameter to a smaller value and override it with a larger ASSIZEMAX setting in the RACF Profile.

Likewise, the MAXTHREADS value can also be overridden by the THREADMAX value specified on the user's RACF USS segment.

In general, it isn't a good idea to run the WebLogic Server startup script from an OMVS shell since the TSO region size will be used. Usually TSO regions are only 4MB in size and the WebLogic Server will very quickly run out of memory. A better approach is to start the WebLogic Server instance using a JCL (Job Control Language) procedure or via a Telnet session. An example of a JCL procedure to start WebLogic Server was included in our second article. One suggestion is to use a Telnet session when configuring the WebLogic Server after installation and during development, then create a JCL program for server startup when ready for production deployment.

VM GUEST OPTIONS

The most important settings for a virtual machine are:

- **The virtual machine size:** This has already been defined with a base of 512MB.
- **The execution class**
- **The share of processor resources WebLogic will receive:** It's a good idea to assign the virtual machine a relative share setting for CPU resources so it doesn't starve other virtual machines.
- **The use of the z/VM Guest LAN to support WebLogic clusters:** This option provides an in-memory LAN segment that WebLogic instances can use to communicate throughout the cluster.

LEGACY APPLICATIONS

One of the key advantages of deploying WebLogic Server on the mainframe is the proximity to the underlying business data and information. There are a number of connectors for mainframe applications that enable calls to legacy sys-

tems to be handled in a very efficient manner. Many of these options for legacy integration were outlined in the second article in this series, including the ShadowDirect adapters available from Neon Systems. In the next article we'll outline the various options for mainframe application integration, including Web services.

Regardless of the adapter or connectivity option used, the configuration options for that adapter should be reviewed. This is particularly important when WebLogic Server and the legacy application are on the same platform, since configuration options may provide an extra performance boost.

JAVA VIRTUAL MACHINE

A number of parameters affect the performance of the Java Virtual Machine (JVM) on the mainframe. The first item to review is the minimum and maximum heap size. This setting controls how often the garbage collector runs. Contrary to popular belief, setting the JVM's heap size too high can in many cases be as bad as setting it too low.

Unless you have detailed knowledge of the application running in WebLogic and how it uses memory, the only way to determine the optimum minimum and maximum heap size values is by trial and error. Setting the heap size too small will result in constant swapping; too large will result in inefficiencies for garbage collection and resource utilization. A good recommendation is to start with minimum and maximum heap size settings of 256MB each. The minimum heap size is set with the "-Xms" option when starting WebLogic Server; the maximum heap size is set with the

"-Xmx" option. Values can then be adjusted based on how the application performs.

The JVM heap memory will be allocated immediately during the server startup above the 16MB line. However, overrides in the JES Job exit and region size on the WebLogic JCL will limit the actual amount of memory allocated, so always make sure the server gets the size specified.

APPLICATION CODE

There is no right or wrong way to code programs, but there are some generally accepted best practices when coding any Java or J2EE application. These practices apply to applications deployed on the mainframe just like other platforms. It's a good idea to review the code and make sure these practices are enforced and followed. A number of sources, such as *BEA WebLogic Developer's Journal*, cover many of these best practices in great detail. In particular, be aware of things like multithreaded servlets, large objects, very granular Enterprise JavaBeans, etc. Although you may not have the luxury of changing the code, particularly when using packaged applications, you can often configure WebLogic Server to tolerate them. For example, it's generally a good idea to isolate certain components into another instance of WebLogic Server and allow the runtime execution to resolve the actual deployment of the components.

PERFORMANCE TOOLS

Once a test environment has been established, the initial load test will form the baseline for future tuning efforts. It is key to have some quantitative way to measure performance. On the mainframe

many performance measurement tools exist, such as Wily's Introscope. With tools such as these the actual internals of WebLogic Server and the business application can be monitored, both in QA and production mode. For example, Introscope collects statistics and metrics in a SQL database. This information can be used for performance analysis and capacity planning, and as a way to compare changes made to the application and the underlying server configuration, such as determining whether increasing the heap size in the JVM actually improves performance.

The vmstat command is another useful tool. Results from this command can be written to a file via the pipe utility during a load test in the QA environment. The vmstat command will display a number of runtime resources, including:

- Paging rates
- Task status
- Memory usage
- CPU times

With this information problems can be identified very quickly. Each of these will assist in pinpointing areas where a potential performance bottleneck might exist.

Customer Examples

BEA has a number of customers running WebLogic Server on the mainframe, covering a broad range of business situations and environments. In this article we have selected three situations – application redeployment, new application deployment, and deployment in a heterogeneous cluster. Each of these examples discusses an actual customer deployment. Together, the examples span both z/OS and z/Linux customers, and represent a general survey of how WebLogic Server on the mainframe is being used to solve critical business problems.

APPLICATION REDEPLOYMENT

In the first case, the customer had an existing application deployed using WebLogic Server on a UNIX server. Unfortunately, this particular application could only support a small number of concurrent users. To meet the requirements dictated by the business unit the customer had to add UNIX servers, increasing hardware and administrative costs, as well as increasing the complexity of the production deployment. The goals in moving to the mainframe were: (1) to exchange the hardware platform without making any code or architecture changes to the application, (2) to move the application closer to the data accessed, and (3) to compare the performance in a mainframe environment with the performance on UNIX hardware.

Existing Deployment

The existing application was based on J2EE standards and deployed on WebLogic Server v6.0 using JDK 1.3.0. Several connectors were used to access data from legacy applications. All of these were written in Java, which enabled the same connectors to be used when the application was redeployed on the mainframe.

Results

The application was deployed to WebLogic Server on the mainframe without making any changes to the design or the application. The existing data connectors were utilized on the mainframe. When the performance tests were run on the new deployment platform, a single WebLogic Server instance achieved five times

LOOK WHAT'S COMING NEXT MONTH

The Power of EJB QL Subqueries in WebLogic Server Version 7

by Thorick Chow

A look at the use of some of the new EJB QL Query Language features that went into release 7.0, with a focus on the support for SubQueries.



WebLogic Portal: Understanding the Advisor Framework

by Dwight Mamanteo

First in a series that will teach you all you need to know about the different frameworks in WebLogic Portal



WebLogic on the Mainframe: Web Services

by Tad Stephens and Eric Gudjon

Tad and Eric finish their series with a discussion of how to Web service-enable existing mainframe applications and data using WebLogic Server.



WebLogic Web Services Security

by Anbarasu Krishnaswamy

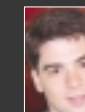
This article takes an "under-the-hood" look at WebLogic's Web services security. As more and more customers adopt Web services, they need to understand how Web services can be secured and the authentication mechanism used to keep the Web services open and support multiple client types.



Sam's Soapbox

by Sam Pullara

Sam returns to WLDJ with a look at managing complexity: BPM and JMX administration and management



WLS Certification: A Final Review

by Dave Cooke

The last in Dave's series on the WebLogic Server 6.0 certification test. Dave looks at administration, and offers a final review test that wraps up all the topics.



Once you're in it...

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

reprint it...

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

Re Prints

SYST-CON MEDIA

WLDJ ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
Altaworks	www.altaworks.com/wlpaper	603-598-2582	52
BEA	http://dev2dev.bea.com/useworkshop	800-817-4BEA	2
Borland	http://info.borland.com/new/javasolutions/92115.html	831-431-1000	41
Covasoft	www.covasoft.com/today's_tech	888-963-2682	19
Dirig	www.dirig.com/illusion/weblogic	603-889-2777	9
EnginData Research	www.engindata.com		49
Hewlett-Packard	www.hp.com/go/developers	650-857-1501	51
Intel	www.intel.com/ad/bea	408-765-8080	3
Panacya	www.panacya.com	877-726-2292 x3344	17
Precise Software	www.precise.com/wldj	800-310-4777	15
Rational Software	www.rational.com/offer/bea	800-728-1212	11
Sitraka JClass Server/Views	www.sitraka.com/jclass/wldj	800-663-4723	21
Sitraka PerformaSure	www.sitraka.com/performance/wldj	800-663-4723	25
Sun Microsystems	www.sun.com/servers/entry/beapromo3	800-765-9200	29
SYST-CON Publications	www.syst-con.com	888-303-5282	34
WebLogic Developer's Journal	www.syst-con.com/weblogic	888-303-5282	35
Web ServicesEdge Conference	www.syst-con.com	201-802-3069	31, 37-38
Web Services Journal	www.wsj2.com	888-303-5282	48
Wily Technology	www.wilytech.com	888-GET-WILY	4

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

the concurrent user load. In addition, this load was achieved while cutting the response time approximately in half.

Summary

In this case the customer was able to utilize the portability of J2EE applications with WebLogic Server, increase application performance with lower response time, lower administrative costs, reduce complexity, and redeploy the application to a new hardware platform without requiring modification to application components or the design. This gave the customer the freedom to choose the hardware platform providing the desired quality of service for their application. In addition, the ease with which the application was deployed to the mainframe suggests that consolidation from a number of UNIX servers to the mainframe is achievable.

NEW APPLICATION DEPLOYMENT

In many cases, customers will decide to develop and deploy new applications on the mainframe to leverage the quality-of-service features available, such as the WorkLoad Manager. A particular benefit found during numerous evaluations with customer systems is that, as the workload increases, the overall responsiveness of the application does not vary widely. Deploying such

services on the mainframe, allocates resources efficiently, allowing customers to plan and predict performance accurately.

In this particular case, several new applications were designed and developed specifically for deployment on the mainframe. The business unit had specified that the applications must be highly available and able to support a large number of concurrent users. The customer determined that deploying the same applications in a cluster of distributed servers would require many more servers, increasing complexity, and in many cases a single server would be needed for each application. By deploying on the mainframe the customer was again able to lower operational and administrative costs, reduce application complexity, and consolidate a number of unique server instances on a single mainframe. Utilizing the WorkLoad Manager gave the customer the necessary degree of application availability while effectively utilizing the underlying resources.

CONTINGENCY DEPLOYMENT

One advantage of WebLogic Server, regardless of the underlying hardware, is the unique clustering technology. Clustering provides application redundancy and failover in a distributed environment. WebLogic Server clustering allows heterogeneous

hardware servers running the same application to be combined within a single WebLogic cluster. In a recent case, a customer decided to use WebLogic Server instances running on the mainframe as backup nodes for the WebLogic Servers running on UNIX in a separate data center.

Background

The customer had already deployed a number of UNIX servers running WebLogic Server. In the event the UNIX servers experienced some critical failure the mainframe running WebLogic Server would assume a portion or all of the workload. Although the UNIX and mainframe hardware were located in separate data centers, the business need for highly available applications required that these heterogeneous platforms be clustered.

Design

The WebLogic cluster included both the WebLogic Server instances running on the UNIX servers and on the mainframe. An HTTP proxy server running in the network DMZ routed initial session requests to the UNIX servers. Using replication partners, session replication was designated to the WebLogic Server deployed on the mainframe. Sessions were persisted on the primary and secondary servers in memory.

Results

In this case, by including both UNIX and mainframe WebLogic Server instances located in separate data centers the customer delivered redundancy between data centers without affecting the end user. The customer achieved high availability and reliability, regardless of the underlying hardware platform, and leveraged advanced clustering features such as in-memory session persistence and replication groups in a heterogeneous environment. Regardless of planned outages or critical events, the application continued processing without interruption.

Conclusion

We've examined a number of performance and tuning tips for planning a WebLogic Server deployment on the mainframe. We've provided configuration and initial settings and outlined a plan for achieving optimal application performance. We've also detailed three scenarios where WebLogic Server is currently in use on the mainframe. In the final article in this series on WebLogic Server and the mainframe we will examine many of the options for data and application integration, including how to Web service-enable applications by using WebLogic Server on the mainframe. Be there or be square! 🍷

WebServices JOURNAL
.NET J2EE XML

LEARN WEB SERVICES. GET A NEW JOB !

Subscribe today to the world's leading Web Services resource

Get Up to Speed with the Fourth Wave in Software Development

- Real-World Web Services: XML's Killer App!
- How to Use SOAP in the Enterprise
- Demystifying ebXML for success
- Authentication, Authorization, and Auditing
- BPM - Business Process Management
- Latest Information on Evolving Standards
- Vital technology insights from the nation's leading Technologists
- Industry Case Studies and Success Stories
- Making the Most of .NET

- Web Services Security
- How to Develop and Market Your Web Services
- EAI and Application Integration Tips
- The Marketplace: Tools, Engines, and Servers
- Integrating XML in a Web Services Environment
- Wireless: Enable Your WAP Projects and Build Wireless Applications with Web Services!
- Real-World UDDI
- Swing-Compliant Web Services
- and much, much more!

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

Subscribe online at www.wsj2.com or call 888 303-5282

The Best .NET Coverage Guaranteed!

engindata RESEARCH
www.engindata.com

SYS-CON MEDIA

SYS-CON Media, the world's leading technology publisher of developer magazines and journals, brings you the most comprehensive coverage of Web services.
 *Offer subject to change without notice

Chart Your Course to Success in IT...

...order your copy of **Java Trends: 2003**

Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?

EnginData Research has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.

engindata RESEARCH
www.engindata.com

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of EnginData Research. Order your copy of the 2002-2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.

Here's just a sample of the critical data points the Java report delivers...

- Current and projected usage of languages and platforms
- Multiple rankings of hundreds of software vendors and tools
- Types of applications being developed
- Databases being used
- Purchasing patterns
- Company size
- Development and purchasing timelines
- Perceptions and usage of Web services
- Preferred Java IDE
- J2EE, J2ME, J2SE usage comparisons

As an IT specialist, EnginData Research focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- Time frame for developing Web services – enabled apps
- Percentage of apps with Web services today
- Sourcing and hosting Web services
- Perceived leaders in Web services tools and solutions

JAVATRENDS 2003

News & Developments

Dirig's Fenway Expands App Server Management Capabilities

(Nashua, NH) – Dirig Software, a developer of award-winning enterprise performance management solutions for e-business, has announced support for BEA Systems' BEA WebLogic Server 7.0. Prospective WebLogic Server 7.0 customers can benefit from Fenway Management Extensions' e-business management capabilities, including Web servers, application servers, and back-end databases. Fenway ensures the performance, availability,



scalability, and transaction integrity of application infrastructures built around an application server.

www.dirig.com

BEA Chooses PANACYA to Help Ensure Superior Performance

(San Jose, CA) – PANACYA Inc., a developer of e-business application performance management products, has announced that BEA Systems will use PANACYA's bAWARE software suite to manage the performance of its customer-facing e-business applications. BEA's decision to use this software was principally based on PANACYA's radically different



approach – distributed intelligent management for distributed applications. This enables BEA to proactively identify and resolve provisioning infrastructure issues before they impact their customers.

www.panacya.com

AdventNet Releases Middleware Manager 4.0 WebLogic Edition

(Pleasanton, CA) – AdventNet,

Inc., a provider of standards-based J2EE management solutions, has announced the immediate availability of AdventNet Middleware Manager 4.0 WebLogic Edition, optimized for advanced JMX (Java Management Extensions) based management of entire Web application ecosystems.

AdventNet Middleware Manager leverages AdventNet's expertise in solutions based on open standards such as JMX and



SNMP. AdventNet has the distinction of having the first Sun-certified independent JMX implementation. The J2EE agent toolkit integrated with Middleware Manager leverages JMX and SNMP standards and facilitates the generation of a custom agent to expose rich advanced management information about the applications deployed on the WebLogic Server.

www.adventnet.com

Altaworks Announces Support for BEA WebLogic Platform 7.0

(Nashua, NH) – Altaworks Corporation, a developer of performance management software for Web enterprises, has announced support for BEA Systems' BEA



WebLogic Platform 7.0. By

supporting this release, Altaworks' Panorama offers BEA customers a purpose-built solution for monitoring and managing the performance of BEA WebLogic Platform 7.0, including patent-pending capabilities to determine probable causes of performance slowdowns throughout an entire Web enterprise.

Altaworks Panorama software is purpose-built using Java technology to help companies monitor, analyze, and ensure optimal performance of distributed applications in all stages of the application life cycle, including development, quality assurance/testing, and production. www.altaworks.com

BEA Expands Developer Offerings

(San Jose, CA) – Continuing to expand its developer ecosystem, BEA Systems, Inc., an application infrastructure software company, has announced the BEA dev2dev Online

Membership Program, a free program exclusively for members of

the growing BEA dev2dev community. This program builds on the core BEA dev2dev offerings launched earlier this year with



new premium resources that include discounts on development resources and education opportunities for developers leveraging the BEA WebLogic Enterprise Platform, so they can more quickly and easily build and integrate reliable Web services and enterprise J2EE applications. The dev2dev Online Membership program is available at <http://dev2dev.bea.com/membership/index.jsp>.

BEA dev2dev is a comprehensive developer program devoted to ensuring the success of hundreds of thousands of developers on the BEA WebLogic Enterprise Platform. BEA dev2dev includes a developer portal, an online membership program, and an integrated package of education and training, tools, support, and

community programs, aimed at developers of all levels of technical knowledge and experience. <http://dev2dev.bea.com>.

OASIS Forms Web Services Security Technical Committee

(Boston) – The OASIS standards consortium has organized a new technical committee to advance the WS-Security specification. WS-Security provides a foundation for secure Web services, laying the groundwork for higher-level facilities such as federation, policy, and trust. Through the open OASIS process, providers and users will come together to extend the functionality of WS-Security, which was originally published by IBM, Microsoft, and VeriSign.

BEA Systems, Blockade Systems, Commerce One,



divine, Documentum, Fujitsu, Intel, IBM, IONA, Microsoft, Novell, Oblix, OpenNetwork, Perficient, SAP, SeeBeyond, Sonic Software, Sun Microsystems, TIBCO, VeriSign, webMethods, XML Global, and other OASIS members will collaborate on advancing the WS-Security specification.

www.oasis-open.org



Hewlett-Packard

www.hp.com/go/developers

Altaworks

www.altaworks.com/wlpaper