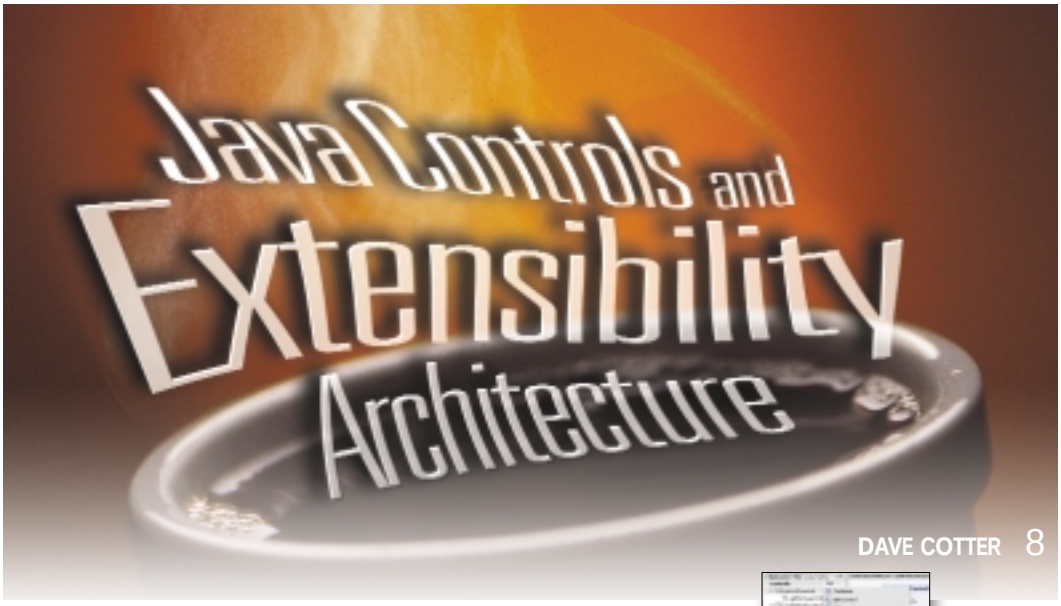
MOVING AHEAD FOR THE 21ST CENTURY

BEA WebLogic DEVELOPER'S JOURNAL

SEPTEMBER 2003 - Volume:2 Issue:9



Informatica

www.informatica.com

The Whole Package

BY SEAN RHODY
MANAGING EDITOR

In the battle over application servers, I've got good news and bad news. The good news is that the battle is over. The bad news is that everybody lost. And by that I mean something a little different from what you might think.

Application servers are no longer a hot property. Yes, you still need them, and they still form the backbone of a J2EE shop's infrastructure. But the days when you selected a platform on the abilities of a particular application server are over. Part of this is due to the relative maturity of the J2EE specification, which has grown into something that developers can now truly code to regardless of platform choice.

Another part of this is the consolidation that has occurred in the marketplace over the past few years. Gone are the days when there were 30 application-server vendors trying to convince you that their clustering was better or their JMS implementation outperformed all others. There's BEA, there's IBM, and then there is low-cost, including Sun, Oracle, and JBoss. And really, only the first two are valid choices. Okay, only the first one is a valid choice.

Today the selection is made on more than just app server functionality – it's on the total package. Companies are evaluating the entire vendor platform in making decisions. Additional features and differentiators are what lead CIOs and CTOs to make the big decision, or to endorse the decision their key architects agree upon.

That's why the release of WebLogic Platform 8.1 from BEA is generating so much excitement. The new platform features a much greater degree of integration of products than previous versions, as well as enhanced support for Web services.

Probably the biggest news is the latest revision of WebLogic Workshop. Aimed at corporate developers, this Visual Studio killer shows real promise in accelerating application development and moving J2EE into the realm of application assembly instead of programming.

One of the biggest challenges in corporate development is that Java, for all its power, is still more complex to program in than VB. And until Workshop, the IDE makers concentrated on optimizing things for hard-core coders, excluding those who don't have either the time or the inclination to get into all of the details of the J2EE specification. There will always be a need for talented programmers who know Java inside and out, but sometimes the corporate mission requires a quick hit without detailed knowledge. That was lacking in the J2EE world until Workshop, which turns EJBs, JDBC connections, and Web services into components that can be wired together to create full-featured applications with minimal coding. Add to it awareness of other parts of the platform, and you have a powerful, easy-to-use IDE with great breadth of features.

• • •

It gives me great pleasure to introduce Joe Mitchko (see page 6) as the new editor-in-chief of ***WebLogic Developer's Journal***. Back in the early '90s, Joe was the person who introduced me to Java, and he and I have worked together on several BEA WebLogic projects. Joe is a dedicated professional with almost 20 years of industry experience, and his insights into the intricacies of J2EE development will be vital in guiding *WLDJ* over the coming years.

AUTHOR BIO...

Sean Rhody is the managing editor of *WebLogic Developer's Journal* and editor-in-chief of *Web Services Journal*. He is a respected industry expert and a consultant with a leading consulting services company.

CONTACT: sean@sys-con.com

BEA WebLogic DEVELOPER'S JOURNAL

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

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JOE MITCHKO

MANAGING EDITOR
SEAN RHODY

PRODUCT REVIEW EDITOR
JASON SNYDER

EXECUTIVE EDITOR
GAIL SCHULTZ

EDITOR
NANCY VALENTINE

ASSOCIATE EDITORS
JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR
JENNIFER VAN WINCKEL

WRITERS IN THIS ISSUE
KINGSUM CHOW, LEWIS CIRNE, DAVE COTTER, WILLIAM COX, GIM DEISHER, PETER HOLDITCH, JOE MITCHKO, SEAN RHODY, MARK ROTHERA, VITALIY STULSKI, HUB VANDERVOORT

SUBSCRIPTIONS
For subscriptions and requests for bulk orders, please send your letters to Subscription Department.
SUBSCRIPTION HOTLINE:
888-303-5282
Cover Price: $8.99/Issue
Domestic: $149/YR (12 Issues)
Canada/Mexico: $169/YR
Overseas: $179/YR
(U.S. Banks or Money Orders)

PRESIDENT AND CEO
FUAT KIRCAALI

VP, BUSINESS DEVELOPMENT
GRISHA DAVIDA

SENIOR VP, SALES & MARKETING
CARMEN GONZALEZ

PRODUCTION CONSULTANT
JIM MORGAN

GROUP PUBLISHER
JEREMY GEELAN

ART DIRECTOR
ALEX BOTERO

ASSOCIATE ART DIRECTORS
LOUIS F. CUFFARI • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR
TAMI BEATTY

VP, SALES & MARKETING
MILES SILVERMAN

ADVERTISING SALES DIRECTOR
ROBYN FORMA

DIRECTOR OF SALES AND MARKETING
MEGAN RING-MUSSA

ADVERTISING SALES MANAGER
ALISA CATALANO

ASSOCIATE SALES MANAGERS
CARRIE GEBERT • KRISTIN KUHNLE

PRESIDENT, SYS-CON EVENTS
GRISHA DAVIDA

CONFERENCE MANAGER
MICHAEL LYNCH

REGIONAL SALES MANAGER
JAMES DONOVAN

FINANCIAL ANALYST
JOAN LAROSE

ACCOUNTS RECEIVABLE
KERRI VON ACHEN

ACCOUNTS PAYABLE
BETTY WHITE

VP, INFORMATION SYSTEMS
ROBERT DIAMOND

WEB DESIGNERS
STEPHEN KILMURRAY • CHRISTOPHER CROCE

ONLINE EDITOR
LIN GOETZ

CIRCULATION SERVICE COORDINATORS
NIKI PANAGOPOULOS • SHELIA DICKERSON
EDNA EARLE RUSSELL

JDJ STORE MANAGER
RACHEL MCGOURAN

EDITORIAL OFFICES
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
BEA WebLogic Developer's Journal (ISSN# 1535-9581) is published monthly (12 times a year)
Postmaster: Send Address Changes to
BEA WEBLOGIC DEVELOPER'S JOURNAL,
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

WORLDWIDE NEWSSTAND DISTRIBUTION
Curtis Circulation Company. New Milford, NJ

FOR LIST RENTAL INFORMATION:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

SYS-CON MEDIA

The 21st Century Developer

BY JOE MITCHKO

THE ABILITY TO MOVE FAST IS HERE NOW

Imagine for a moment what it would be like to be a 21st-century software developer sitting in your cubicle at work. All right, I know that we are already a few years into the century, so let's push it up, say, 50 years. Okay, that's better. First of all, you may notice that things have changed a bit. A nice three-dimensional hologram screen surrounds me completely. I can just move stuff around by grabbing it and I can even float around in a three-dimensional virtual space? I like that. And look at that, Dell is still in business. Awesome, dude!

But wait, I'm a developer right? Where is my usual stuff? My IDE, compilers, and debugging tools – all seem to be missing. Go through that virtual revolving door over there? Ah, I see. Whoa, what an awesome-looking machine! As far as the eye can see, there appears to a living digital organism, in constant motion, with lines and pipes full of messages going everywhere. What's that you say? I'm looking at what? This is the entire corporation, complete with all of its processes, running in real time? Very cool! I smell money, so that must be finance over there.

Okay, though, what application do I need to change? We don't do that anymore, we just modify the underlying business process and that's it? And depending on your responsibility and role in the corporation, the system will provide you with everything you need to do your job, including data entry, workflow, etc.?

This may sound like 22nd-century stuff, but if certain industry visionaries are correct, we're already headed down that path with the advent of new standards based on Web services and business process modeling. And the vision of what development would be like in the future is precisely what came to mind the first time I was introduced to BEA Workshop.

With Workshop, programming the underlying J2EE components and the multitude of configuration settings for a Web service or application is done for you automatically – almost as quickly as you can manipulate controls and connectors in a graphical environment. And you can deploy the change to your production environment with the push of a button. All that's really left to do is program in the business logic, and even that's about to change with the release of WebLogic 8.1.

When you add the additional capabilities of BEA WebLogic Integration and Liquid Data, developing and maintaining business rules (the underlying heart of the business process) will be just as easy. And as the industry gains wider support for standards such as BPEL, Integration Server will be ready to go when it comes time to import and export business process information.

You see, we need to understand that constantly changing business requirements are a normal and healthy thing – from the perspective of the business, that is. In order for a business to compete and survive in today's global economy, where the business value chain extends out to a web of external business partners, it will need to constantly tweak its business processes, and in an increasing number of cases, develop customized processes specific to a customer's requirements.

As developers, it is our insistence that business requirements stay static (while we take time to build applications) that tends to get in the way. What this means for today's IT shop is this – we need to be able to make rapid-fire application changes that react to business process change, and do it in as close to real time as possible. And this is precisely how I see the WebLogic 8.1 platform providing a path to the future.

Okay, let me grab a cup of coffee and head back to my cubicle of the future. Wait a minute! You mean to tell me that caffeine is now an illegal substance!

> "We need to understand that constantly changing business requirements is a normal and healthy thing"

AUTHOR BIO...

Joe Mitchko is a senior technical specialist for CSC Consulting in New York and was product review editor and a contributing writer for *Web Services Journal*.

CONTACT: joe@sys-con.com

WebLogic Workshop Java Controls and Extensibility Architecture

BY **DAVE COTTER**

A RICH MODEL TO EASILY INTEGRATE INTO THE DEVELOPMENT ENVIRONMENT.

AUTHOR BIO...

Dave Cotter is a director of product management on the WebLogic Workshop product team.

CONTACT...

dcotter@bea.com

One of the most exciting aspects of BEA WebLogic Workshop 8.1 is the ability for developers and ISVs to extend the Workshop development environment by developing custom Java Controls, IDE Extensions, application templates, or TagLibrary extensions. In this article we start with a brief overview of WebLogic Workshop 8.1 and then take a high-level tour of these extensibility points. This article has been assembled from several sources on the Workshop product team. Workshop extensions development kits are available at http://dev2dev.bea.com/products/wlworkshop81/

WebLogic Workshop 8.1 Overview

WebLogic Workshop 8.1 is the IDE and framework for creating J2EE applications for the BEA WebLogic Platform 8.1. It's optimized for all developers to rapidly create, test, and deploy Web applications, XML Web services, EJBs, portals, and business processes on BEA WebLogic Platform 8.1. Simply put, Workshop makes the hard things easy and the sophisticated things possible.

WebLogic Workshop is both an IDE and a runtime framework. Its visual development environment and programming model is based on intuitive concepts such as drag-and-drop controls, methods, and properties to enable event-based development, eliminating the need for developers to master complex J2EE APIs and object-oriented

programming disciplines. The IDE produces standard Java files with additional annotations to specify the appropriate runtime application behavior. These annotations enable the Workshop runtime framework to automatically generate the J2EE infrastructure components, thereby abstracting the user from the low-level infrastructure plumbing that would otherwise be required.

The visual development environment produces standard Java files with additional annotations inserted by Workshop (for example, when the developer sets properties or adds controls) to specify the appropriate runtime application behavior. These annotations enable the Workshop runtime framework to automatically generate the J2EE infrastructure components, thereby abstracting the user from the low-level infrastructure plumbing that would otherwise be required.

Applications – whether they are Web services, Web applications, portals, or workflows – share a common process of assembling and wiring together components that encapsulate business logic or resources with additional code, workflow logic, personalization information, and so on. Application assembly and component reuse are key constructs within WebLogic Workshop.

WebLogic Workshop Extensibility Points

In addition to simplifying the creation of J2EE applications, Workshop offers a rich extensibility model for developers and ISVs to integrate their products and services directly into the development environment. Using Java Controls, Templates, IDE Extensions, and TagLibrary extensions, developers have broad flexibility in working with WebLogic Workshop. Some example Workshop extension and application integration ideas include:

- Custom application designers and windows that reside in Workshop's IDE
- Custom controls that interface with back-end resources including databases, systems, applications, and business logic
- Tool bar icons and custom menus that launch external helper applications
- New project and file types not native to Workshop
- JSP Tag Library extensions for use in Web applications

With this as the backdrop, let's look at these extensibility points in more detail.

WebLogic Workshop 8.1 Java Controls

BEA WebLogic Workshop Java Controls provide developers with the ability to easily connect with existing data, systems, applications, and business logic. Controls are visual components with events, methods, and properties that handle all of the details of connecting to an external resource or piece of business logic. Developers interact with controls by handling events, calling methods, and setting properties. In addition to the suite of built-in Controls (see Figure 1), Workshop 8.1 also makes the Controls model extensible so that all users (including ISVs) can build custom Controls that seamlessly plug into the Workshop integrated development environment.

Creating Java Controls

To create a custom Java Control, developers use familiar visual designers to specify the interface (i.e, the methods and events supported), set properties to dictate runtime behavior, and write the business logic using procedural Java code to implement these methods. Developers build the control(s) in a separate Workshop "Control Project" within an application. When the Control Project is built, all of the controls in the project are compiled into a single JAR file. This JAR is self-contained and can be moved to any other Workshop application by adding it to the application's Library folder.

Workshop also helps guide the Control user through the tasks of configuring the Control's behavior and appearance within an application. Control authors can provide either a simple Swing-based JPanel wizard or a multistep wizard presenting a set of configuration choices in terms the user understands.

Java Controls expose a set of simple properties by which a user can implement advanced runtime functionality such as asynchronous communication, security roles, life-cycle events, transaction support, and so on. Authors of custom controls have the freedom to populate the Properties sheet for the benefit of the controls' users.

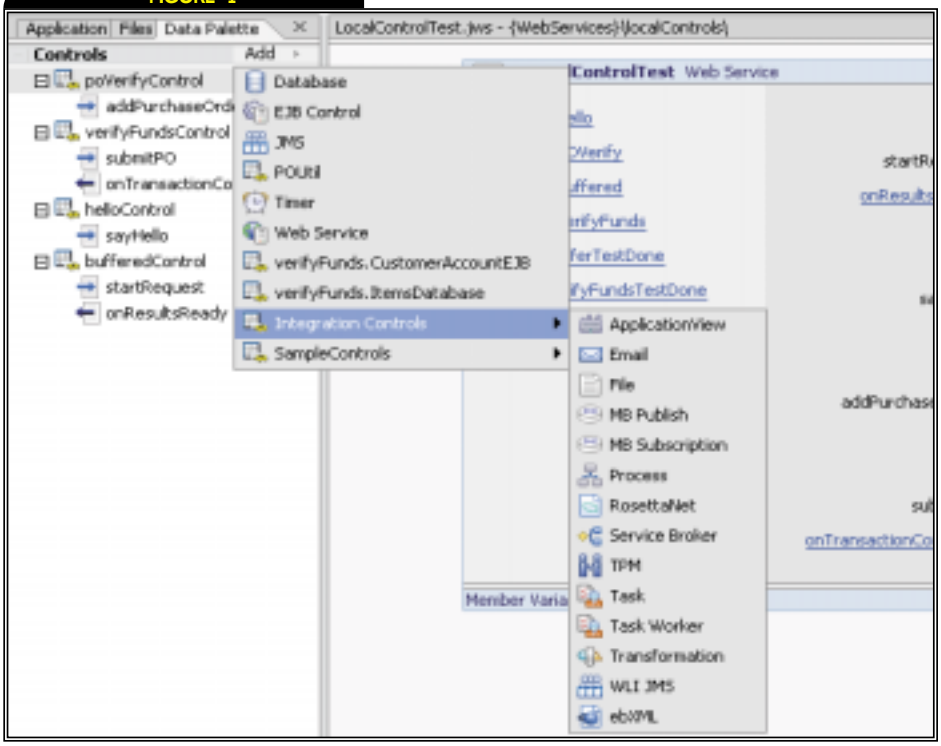
Packaging and Distributing Java Controls

Controls are easily packaged and distributed as standard JAR files. Users consume controls by importing JARs into their application Library folder (see Figure 2). Controls then automatically appear on the design palette of the user, ready for incorporation into an application.

Application and Project Templates

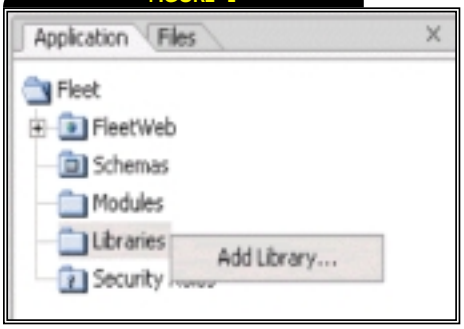
WebLogic Workshop ships with a project and application template mechanism that is part of the application subsystem. Templates allow development teams to start with an appropriate architecture as part of the initial project structure by prepopulating an application or project with the required supporting files and resources (i.e., JSPs, EJBs, JAR files, images, etc.).

FIGURE 1



Built-in Java Controls

FIGURE 1



JARs in the application Library folder

They offer a way for system architects and ISVs to package and distribute best practices and starting points for developers.

Templates are stored in the weblogic81\workshop\templates directory and are packaged as zip files. A template can be made up of Java Controls, EJBs, JSPs, images, and/or any other file. They are loaded into an IDE based on an XML template definition file. Template zip files are read the first time the New Application, New Project, Add Project, or Install menu is opened.

Project Templates

WebLogic Workshop supports two types of templates: application and project. A project template can be created for populating a single new or existing project of a specific project type within Workshop. A project template may also be extended by other project templates and referenced by application templates.

Application Templates

An application template can be used for populating a new or existing Workshop application. Application templates can contain any number of elements, including project templates. Project templates defined inside the application template will not appear in the New Project and Add Project dialog and may only be used when creating this application.

Listing 1 is an example of the template definition file associated with creating Web applications.

WebLogic Workshop 8.1 IDE Extensions

There are several levels of depth by which to extend and integrate with WebLogic Workshop's IDE; from simple tool bar icons that launch external applications to fully integrated applications with custom windows, menus, and property sheets.

A Workshop extension is composed of the following artifacts, packaged together as a JAR or directory.

- ***XML descriptor:*** extension.xml
- ***Set of Java classes:*** Exposing interfaces, and consuming various services
- ***Resource files:*** Images and string property files
- ***Documentation***

WebLogic Workshop itself is the culmination of several extensions pulled together into one common interface. wlw-ide.JAR is the main Workshop executable, containing the core class files, utility classes, and the mechanism to load extensions, but the interesting code to run the IDE is contained in the required Workshop extensions. WebLogic Workshop defines two types of extensions: required and optional. Required Workshop extensions are:

- Control
- Debugger
- JSP Designer
- Shell
- Source control
- Workspace
- XML
- Source editor

Optional extensions include

- Pageflow
- Process

To cause Workshop to load a specific extension at startup, the extension's JAR file is placed in the extensions' subdirectory below the directory containing the wlw-ide.JAR file that contains the IDE core. An extension may also be fully exploded to directories and class files in the "extensions" subdirectory.

Creating Workshop Extensions

A WebLogic Workshop extension is simply defined as a JAR or a directory that contains at a minimum the extensions.xml, the file that declares an extension's participation in the IDE. By creating a JAR file with the extensions.xml file shown in Listing 2, we can add a new tool bar icon to Workshop. It's that easy.

The % signs in "%sqlEditor.extension.-actionSQLAttributeEditor%" are a means of referring to a localizable string resource and that it's optional.

This extension places a new button on the toolbar with database.gif as its icon. When the new toolbar button is pressed, the SQLAttributeEditor is displayed.

For more complex extensions, the JAR file would contain the Java code that is the extension's implementation, a manifest file that defines the class path, and attributes that reference dependent JARs that need to be available at runtime by the extension.

At startup, the core IDE runtime reads all the extension.xml files, batches them together and ensures that the requested services by each extension are available.

Extensions may define handlers for the <extension-xml> tag found in the extension.xml file. Handlers are associated with a particular ID attribute. All extension.xml files are scanned for fragments contained within the <extension-xml> tag and those fragments are passed to handlers defined for the particular *id* attribute. This mechanism allows extensions to create extendable infrastructure in which other extensions can participate. The handler class is instantiated by the core IDE and is completely responsible for parsing the XML contained in the fragment.

Services

One important aspect of an extension is its ability to expose and consume the various IDE services. A service is essentially a public interface that has a single instance implementation and provides access to functionality in an extension. In addition to handlers for <extension-xml> tags, an extension may declare services that it implements. For instance, the debugger extension defines a debugger service that, among other things, provides a method for setting a breakpoint. The shell provides a document service with a method for opening a document. Services are consumed by the extensions Java classes and registered with the system using special tags in the extensions.xml file. Many services have associated <extension-xml> tag handlers that allow extensions to add functionality to the service. Some of the most widely used services include:

- ***Resource Service:*** Provides access to resources such as icons, images, and localizable strings
- ***Frame Service:*** Allows extensions to specify the extensions docking layout
- ***File Service:*** Provides a set of services for file system access and manipulation
- ***Server Service:*** Provides a set of services for accessing the WebLogic Server
- ***Document Service:*** Provides a means to supply an abstract document interface for files that are part of the application project.
- ***Action Service:*** Provides methods for adding and manipulating menu and toolbar items

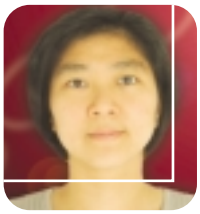
continued on page 16

Quest Software

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

SPECjAppServer2002 Performance Tuning

PERFORMANCE MAY BE A CAKE – YOU NEED TO CHECK ALL THE LAYERS



BY **KINGSUM CHOW & GIM DEISHER**

AUTHOR BIOS...

Kingsum Chow is a senior performance engineer working with the Managed Runtime Environments group within the Software and Solutions Group (SSG). He has been involved in performance modeling and optimization of middleware application server stacks, with an emphasis on J2EE and Java Virtual Machines. Kingsum holds a PhD in computer science and engineering from the University of Washington.

Gim Deisher is a senior software performance engineer working with the Software and Solutions Group at Intel Corporation. She holds an MS in electrical engineering from Arizona State University.

CONTACT...

kingsum.chow@intel.com
gim.l.deisher@intel.com

This article discusses the best known methods for tuning the performance of the BEA WebLogic application server running the SPECjAppServer2002 benchmark on Intel architecture platforms. We describe a top-down, data-driven, and closed-loop approach to performance tuning, and touch on key advantages of BEA WebLogic that improve the performance of J2EE workloads.

Introduction

Java has become increasingly important in server-based applications. Consequently, standardized, robust, and scalable application support frameworks have become critical. Java 2 Enterprise Edition (J2EE) addresses this need, providing a comprehensive specification for application servers, including componentized object models and life cycles, database access, security, transactional integrity, and safe multithreading. One such application server is the BEA WebLogic application server. SPECjAppServer2002 is the most recent client/server benchmark for measuring the performance of Java Enterprise application servers using a subset of J2EE APIs in a Web application with a focus on Enterprise JavaBeans (EJB) performance.

In this article we'll examine the performance of SPECjAppServer2002 running on WebLogic and Intel architecture server platforms. We describe an iterative, data-driven, top-down methodology (see Figure 1), and the tools needed to systematically improve system performance.

At the system level, we identify performance and scalability barriers such as input/output (I/O), and operating system and database bottlenecks; and techniques to overcome those barriers. At the application level, we'll discuss application design considerations and application server tuning. At the machine level, we'll discuss Java Virtual Machine (JVM) tuning.

Performance Tuning Methodology

Application server configurations frequently involve multiple interconnected computers. Given the complexity involved, ensuring an adequate level of performance in this environment requires a systematic approach. There are many factors that may impact the overall performance and scalability of the system. Examples of these factors include application design decisions, efficiency of user-written application code, system topology, database configuration and tuning, disk and network input/output (I/O) activity, operating system (OS) configuration, and application server resource throttling controls.

We first apply existing generic best-known methods (BKM) to the system under test and obtain initial performance data. The initial performance data establishes a baseline for us to move forward by applying changes to tune the system and measure performance enhancements arising from these tuning efforts. The steps in the iterative process, shown in Figure 2, are:

1. ***Collect data:*** Gather performance data as the system is exercised using stress tests and per-

formance monitoring tools to capture relevant data.
2. ***Identify bottlenecks:*** Analyze the collected data to identify performance bottlenecks.
3. ***Identify alternatives:*** Identify, explore, and select alternatives to address the bottlenecks.
4. ***Apply solution:*** Apply the proposed solution.
5. ***Test:*** Evaluate the performance effect of the corresponding action.

Once a given bottleneck is addressed, additional bottlenecks may appear, so the process starts again by collecting performance data and initiating the cycle, until the desired level of performance is attained.

Data collection utilizes the following types of tools:

- ***System monitoring:*** Collect system-level resource utilization statistics such as CPU (e.g., % processor time), disk I/O (e.g., % disk time, read/write queue lengths, I/O rates, latencies), and network I/O (e.g., I/O rates, latencies). Examples of tools used to measure these quantities are "perfmon" on the Microsoft Windows OS, and "sar/iostat" on the Linux OS.
- ***Application server monitoring:*** Gather and display key application server performance statistics such as queue depths, utilization of thread pools, and database connection pools. For example, BEA's WebLogic Console can be used to monitor such data.
- ***Database monitoring tools:*** Collect database performance metrics, including cache hit ratio, disk operation characteristics (e.g., sort rates, table scan rates), SQL response times, and database table activity. These may be measured using Oracle 9*i* Performance Manager, for example.
- ***Application profilers:*** Identify application-level hot spots and drill down to the code level. Intel's VTune Performance Analyzer may be used to accomplish this.

Performance Tuning

Before we start tuning the system, a lot of effort can be saved by following currently established BKMs. In this section, we'll look at how we applied BKMs to establish the baseline data. Then we'll describe the iterative approach to tune the system for best performance.

A good source of tuning BKMs can be found from the full disclosures of the publications on the SPEC Web site.

Establish Baseline by Applying Current Best Known Methods

HARDWARE

It's important to ensure that the BIOS settings and the populating of the memory subsystem are done following prescribed norms. Reading and following the system documentation can pay dividends. For example, a platform with 4 gigabytes of memory may perform better with four 1-gigabyte memory cards rather than with one 4-gigabyte memory card. We fully populated the memory banks for the systems under test to eliminate known memory latencies caused by having unfilled memory card slots.

There are several hardware aspects that affect performance, including processor frequency, cache sizes, front-side bus (FSB) capacity, and memory speed. In particular, higher frequency and larger cache lead to improved SPECjAppServer2002 performance. In one study, the performance was improved by 40% when the frequency was increased by 50% and the cache size was doubled.

Network equipment has become relatively inexpensive. We use 1Gbps NICs to reduce the risk that the network becomes a bottleneck.

While focusing on the best performance on the application server, we want to reduce the risk that the database system may become a bottleneck. A high-performance disk array system is used for the database back end. We use eight disks for tables and four disks for logs. We also use raw partitions to avoid the OS file system overhead to access the disks.

OPERATING SYSTEMS

On some Linux systems, the default number of "open files" might be too small for such enterprise Java applications. We increased the limit by adding the following to /etc/sysctl.conf

```
fs.file-max = 65535
```

Similarly, a

```
ulimit -n 65535
```

is added to the application server startup script, or the user's initialization environment (.bashrc).

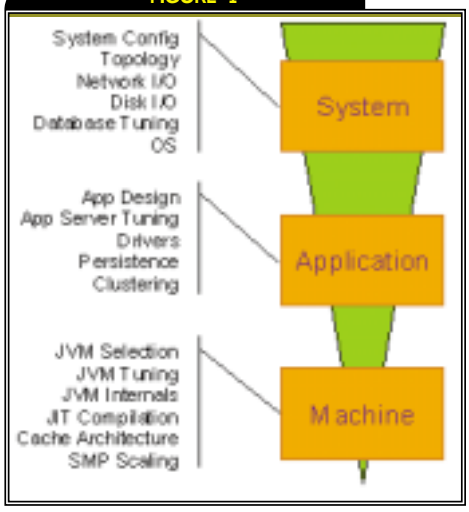
The latest kernel or OS build with known good performance should be used on all systems. Similarly, the most current drivers, for example, the NIC drivers, should also be installed on the application server, database server, and client system.

Once the baseline performance is established, we proceed to the iterative approach to tune the performance at the system, application server, and JVM levels.

For the SPECjAppServer2002 workload, a higher throughput can be obtained by increasing the load (also known as the injection rate) on the system. However, there are response-time requirements that make merely increasing the injection rate overly simplistic. While increasing the injection rate, we need to tune the rest of the system so that adequate response times can be achieved for key transactions.

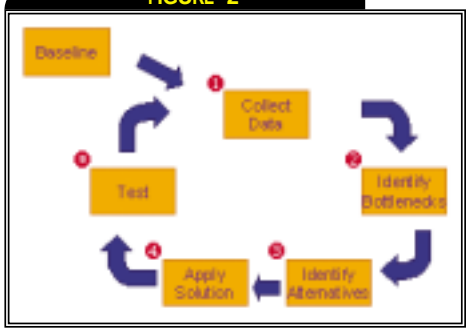
One method we frequently apply is to estimate the likely load on the system that can be supported by scaling a compliant injection rate by CPU utilization to 90%. For example, if we have a compliant run at an injection rate of 100 that consumes 45% CPU, we will increase the injection rate to

FIGURE 1



Performance optimization considerations at the three levels of the top-down stack: system level, application level, and machine level.

FIGURE 2



The iterative, data-driven, top-down process for performance tuning and optimizations.

200 and tune the system so that the response times are compliant.

SYSTEM LEVEL PERFORMANCE

The response time is an important aspect of the SPECjAppServer2002 workload. Thus, lowering the use of resources by the system components will be helpful. During system-level tuning, the main goal is to saturate the application server CPU (i.e., close to 100% utilization). Reaching maximum throughput without full saturation of the CPU is an indicator of a performance bottleneck such as I/O contention, over-synchronization, or incorrect thread pool configuration. Conversely, a high response time metric with an injection rate well below CPU saturation indicates latency issues such as excessive disk I/O or improper database configuration.

Application server CPU saturation indicates that there are no system-level bottlenecks outside of the application server. The throughput measured at this level would point out the maximum system capacity within the current application implementation and system configuration parameters. Further tuning may involve adjusting garbage collection parameters or adding application server nodes to a cluster.

Most components will exhibit a nonlinear response time/throughput behavior. In other words, increasing the throughput will tend to increase the response time, with a disproportionate increase in response time at high throughputs. It is important to size these components such that the required throughput utilization for the component will be relatively low to allow for the response time to be relatively small as well. This is especially important for network capacity, disk capacity, and the capacity of the data bus connecting processors to memory and I/O.

System monitoring tools (described earlier) can be used to track system performance metrics, which can help find bottlenecks. In a multitiered system setup where multiple computers are used, it is important to run these tools on all of the computers.

APPLICATION SERVER–LEVEL PERFORMANCE

The workload itself plays an important factor in performance and it may demand a specific optimal application server configuration. Many parameters can be tuned to optimize for both response times and throughput. Reducing overall response time can often help increase the capacity for further increase in throughput. It is also important to break the response times down into subcomponents, and to further tune the system so that response times of key subcomponents are optimized too.

Many of these tunable parameters are easily accessible from common application servers such as the BEA WebLogic Server. The list of parameters presented here is not exhaustive. It is merely a starting point to tune the performance for your enterprise Java applications. The list includes tuning key application server parameters, and tuning key container parameters. You should bear in mind that these parameters are tuned to reduce response time for key transactions, such as new order and manufacturing, for the workload.

TUNING KEY CONTAINER PARAMETERS

Many application server parameters can be tuned to help an application perform more effectively. The following parameters should be considered for most applications.

- ***Setting a good value for the initial bean pool size improves the initial response time for EJBs:*** They are preallocated upon application server startup.
- ***Setting an optimal value for bean cache size will prevent passivations:*** It increases performance by reducing file I/O activity.
- ***Allocating large enough cache size for the appropriate stateful session bean can potentially improve the throughput:*** For example, you may want to increase the max-beans-in-cache specified for the CartSes EJB and measure the change in performance.

TUNING KEY APPLICATION SERVER PARAMETERS

Many application server parameters can be tuned to enable better sharing and interaction with virtual machines and operating systems. The following parameters should be considered for most applications.

- ***A platform-optimized socket multiplexer should be used to improve server performance for I/O scalability:*** In particular, when a performance pack is available from a vendor it should be used. However, with the emergence of JDK 1.4, this effect has become less significant than before.
- ***The thread pool size should be gradually increased until performance peaks:*** Beware of making this size too big, as a higher number may degrade performance due to unnecessary usage of system resources and excessive context switches.
- ***BEA WebLogic Server supports the notion of multiple queues for transactions:*** You may find a specific distribution of executing threads to optimize for a specific workload. This is particularly important when certain transactions have tight response time limits and more threads for those transactions can be allocated accordingly. The support of multiple queues has a clear advantage over a single queue mechanism for shifting long response time transactions to less critical areas. We found that changing a thread pool size by as small a value as 1 can sometimes yield a big response time improvement.
- ***The database connection pool should be set equal to or larger than the number of available execute threads:*** An execute thread does not need to wait for a connection. For optimistic concurrency the number of connection pools required is actually about 1.5 times the number of available execute threads.
- ***Experimenting with the JDBC prepared statement cache size may yield a configuration that minimizes the need for parsing statements on the database.*** The value should be gradually increased until performance peaks. We started with a value of 100 for SPECjAppServer2002 and did not observe performance gains either increasing or decreasing the value.
- ***Relationship caching and optimistic concurrency are two additional features provided by BEA WebLogic Server:***

JVM-LEVEL PERFORMANCE

Selecting the correct JVM is critical. It is essential to use a JVM that has been optimized for the underlying hardware. The best optimizations for various processor platforms are known, and a Java application needs to rely on the JVM to harness these optimizations.

A JVM can provide configuration parameters to the users to let them identify which techniques the JVM should use for optimal performance of their application. We selected BEA WebLogic JRockit as our JVM as it is highly optimized for both Intel Xeon and Itanium platforms.

The key JVM parameters are in the area of heap management, ranging from the selection of the garbage collection (GC) algorithm and the specification of heap sizes, down to the specifics of thread local allocation sizes and when the space for an

object is cleared. It is usually preferable to set the minimum and maximum heap sizes to be the same to avoid runtime overhead associated with expanding and contracting the heap.

The selected heap size can have a profound effect on performance. It is often desirable to set the heap space as large as possible provided you have enough memory on the system. We use a heap space of 1.5GB for our setup for the Xeon processor family, while we use a heap space of 12GB for our setup for the Itanium processor family as the 64-bit architecture systems allow us to use much more memory to boost the performance.

The BEA JRockit JVM permits alternate garbage collection strategies to be specified. Parallel GC is a good starting option for the SPECjAppServer2002 workload.

While rules-of-thumb can be created and experience can be a guide, there is no real substitute for running a variety of experiments to identify the JVM parameters that work best for a given application. This is especially important for JRockit, which exposes a rich set of parameters for you to squeeze the last drop of performance from your application.

Summary

This article described a top-down, data-driven, and closed-loop approach to boost SPECjAppServer2002 performance. The opportunities to improve performance were examined from the whole system including the software/hardware stack of the system level perspectives, the software applications, and the machine level for the virtual machine as well as the physical hardware. Our research suggested that all layers – not just one or two – of the system stack should be examined for performance bottleneck identification and removal.

Acknowledgments

Jason A Davidson, Ashish Jha, Michael LQ Jones, Tony TL Wang, D J Penney, Kumar Shiv, and Ricardo Morin provided key information for this article.

References

- Arnold, Ken; Gosling, James; Holmes, David. (2000). *The Java Programming Language, Third Edition*. Sun Microsystems, Inc.
- Java Community Process, "Java 2 Platform, Enterprise Edition 1.3 Specification": http://jcp.org/aboutJava/communityprocess/final/jsr058/
- Standard Performance Evaluation Corporation (SPEC): http://www.spec.org/jAppServer2002/index.html
- Chow, K.; Morin, R.; and Shiv, K. (February 2003). "Enterprise Java Performance: Best Practices." *Intel Technology Journal*. http://developer.intel.com/technology/itj/2003/volume07issue01/.
- Intel Corporation, "VTune Performance Analyzer": www.intel.com/software/products/vtune/vtune61
- Patterson, David A.; Hennessy, John L. (1997) *Computer Organization and Design: The Hardware/Software interface*. Morgan Kaufmann Publishers.
- BEA WebLogic Server 8.1: www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/server
- Intel Corporation, "Intel Itanium 2 Processor Reference Manual for Software Development and Optimization": http://developer.intel.com/design/itanium2/manuals
- Intel Corporation. "Intel Pentium 4 Processor Optimization Reference Manual": http://developer.intel.com/design/pentium4/manuals

continued from page 10 PLATFORM

Custom Tag Library Extensions

Tag Library Definition Extension (tldx) files allow developers to integrate their tags into Workshop. Integration services provided by the IDE allow developers access to both the data palette and design palette, error checking in source view, and custom rendering of their tags in the JSP Designer.

To build custom tag libraries for WebLogic Workshop, developers supply a tld/tldx pair, a tag handler JAR, and a tldx handler JAR (currently the tldx JAR must live in the workshop/extensions directory). Associated Help documents may be placed in WebLogic Workshop's Help directory.

Conclusion

BEA WebLogic Workshop 8.1 offers a rich extensibility model for developers and ISVs to integrate their products and services directly into the development environment.

Listing 1

```
<template-definition>
    <project-template id="default"
            type="urn:com-bea-ide:project.type:WebApp"
            extends="base">
        <display
            location="newdialog,importdialog"
            label="Web Project"
            description="Creates a default web application project. By default, the project name will become a part of the URL for the web applications you create in it."
            priority="100"
            categories="Web User Interface" />
        <content type="archive" destination="project" source="netuitemplate.zip" overwrite="true"/>
        <content type="archive" destination="project" source="jsp_resources.zip" overwrite="true"/>
    </project-template>
    <project-template id="base_install" type="urn:com-bea-ide:project.type:WebApp" extends="base" processor="workshop.pageflow.ui.dialogs.wizards.repair.util.ProjectRepairHelper$TemplateProcessor">
        <display
            location="contextmenu"
            label="Web Project Libraries..."
            description="Installs the base set of NetUI / Page flow libraries." />
    </project-template>
```

Listing 2

```
<extension-definition>
<extension-xml id="urn:com-bea-ide:actions">
<action-ui>
<toolbar id="sqltoolbar" path="toolbar/main" priority="2" label="%sqlEditor.extension.actionSQLAttributeEditor%">
  <action-group id="sqledit" priority="10" />
  </toolbar>
  </action-ui>
  <action-set scope="com.bea.ide.lang.control.ControlDocument">
 <action class="sqlEditor.SQLAttributeEditor" label="%sqlEditor.extension.actionSQLAttributeEditor%" icon="images/database.gif">
  <location priority="10" path="toolbar/sqltoolbar/sqledit" />
  </action>
  </action-set>
  </extension-xml>
  </extension-definition>
```

Enterprise Portal Integration and the Enterprise Service Bus

PART 1

BY **MATT ROTHERA & HUB VANDERVOORT**

THE SIMPLE MANAGEMENT OF COMPLEXITY

AUTHOR BIOS...

Hub Vandervoort is vice president of Professional Services for Sonic Software. He has over 20 years of experience as a consultant and senior technology executive in the networking, communications software, and Internet industries. Hub previously cofounded three start-up ventures, including early message-oriented middleware (MOM) leader Horizon Strategies, Inc.,

Matt Rothera is a practice manager at Sonic Software (www.sonicsoftware.com), and works with customers to help plan, design and deploy real-time, service-oriented architectures. With over 15 years of technical experience, Matt has worked with Fortune 100 customers to integrate their services with internal applications, business partners, and enterprise portals.

CONTACT...

hvanderv@sonicsoftware.com
mrothera@sonicsoftware.com

The high interest in business visibility, flexibility, and knowledge management has made portal technology a popular choice for monitoring, searching, and managing business activity across the enterprise. BEA WebLogic Platform provides an attractive platform for building highly dynamic enterprise portals that can aggregate, organize, and present information from multiple back-end systems. Whether the portal is implemented with the feature-rich environment of WebLogic Portal Server, or through an MVC framework such as "Jakarta Struts", the newest release of BEA's 8.1 platform provides many new features that create a powerful portal environment.

However, as enterprise portals begin to aggregate ever-increasing numbers of services across disparate technology domains, geographic boundaries, and even organizational boundaries, portal architects will need to find ways to manage the complexity of the environment. Clearly, building "one-off", reliable, point-to-point integrations between the portal and each service does not scale from a development, deployment, or runtime perspective. There is a need for the concept of a "services network," providing reliable transport, intelligent routing, advanced services management features, and the ability to operate in a highly distributed and federated environment.

Fortunately, a new type of infrastructure known as the enterprise service bus (ESB; see sidebar "The ESB: Your Web Services Network") provides the level of sophistication required for even the most demanding of portal environments. An ESB is a standards-based, service-oriented backbone capable of reliably connecting and coordinating hundreds of application endpoints.

ESBs provide the ideal architecture for enterprises needing to connect a variety of heterogeneous systems distributed across disparate data centers, while still maintaining absolute transactional integrity. Furthermore, it provides several advanced services that are primarily configured through deployment time constructs, shielding portal applications from constantly needing to be revised and redeployed to manage changes on the back end.

Even if you are planning to deploy a portal to integrate systems within a homogenous environment with a handful of back-end services, there are still significant advantages in incorporating

an ESB as the integration network. A comprehensive ESB vendor will provide out-of-the-box management, security, reliability, high performing service requests, native XML processing, complex routing, and transformations as value-added features in the bus. Furthermore, as a standards-based implementation of service-oriented architecture (SOA), an ESB provides the necessary abstraction layer to fulfill the full promises of SOA. The ESB provides the ability to seamlessly relocate the underlying service implementation to another technology, geographic, or organizational domain, without sacrificing the traditional values of homogenous environments: management, security, reliability, scalability, and performance.

In this article, we'll examine two key use cases that demonstrate the power and flexibility of the ESB in integrating highly distributed services with portals and Web applications using BEA WebLogic Platform 8.1. In the next part in this two-part series, we will show you how the ESB can provide additional value to the organization through operational awareness, business activity monitoring (BAM), and service-oriented business process management, allowing the portal to fully harness the full value of the services within the extended enterprise.

Avitek Medical Records: Expanding Its Business Model

The WebLogic 8.1 Platform ships a comprehensive J2EE tutorial known as "Avitek Medical Records," designed to demonstrate all of the core features of the J2EE platform. The tutorial is modeled around the idea of a portal (a J2EE Struts application) that provides the ability for patients, physicians, and administrators to view an aggregated set of medical records. In addition, the application demonstrates the ability to link with external clients through Web services interfaces, providing the ability to interact with the Avitek services without going through the presentation tier.

As a premise for this article, we are going to assume that "business is booming" at Avitek and they have acquired three new sources of medical records to integrate into the portal. To complicate matters, the new sources are geographically distributed across suspect networks, and contain a mixture of technology environments (a variety of J2EE application server vendors, Microsoft .NET, and custom solutions). In fact, Avitek would like an architecture that accommodates change, as they will most likely be acquiring new sources of information incrementally, and migrating the sources to the latest and greatest technologies slowly over time.

Although the architecture for Avitek Medical Records advertises the ability to integrate new sources of medical records using HTTP/Web services, this article will demonstrate some key use cases that indicate that a pure-play Web services environment is not enough. There are issues related to reliability, performance, richness of communications models (request/response, unidirectional, asynchronous services, etc.), quality of service, performance, management, and security that call into question the viability of a pure HTTP/Web services approach. For this reason, we recommend that Avitek modify its architecture to incorporate the notion of an ESB (see Figure 1).

The ESB, based on industry-standard JMS, would provide the enterprise-grade backbone to reliably link services together into a cohesive operating unit, capable of servicing the full range of integration scenarios required by most portals. This article will cover a couple of the many scenarios available once the ESB is put in place:

1. ***Forward Cache:*** The ability to move data from distributed systems close to the presentation tier for low-latency, read-only access to the data
2. ***Federated Query***: The ability to efficiently query multiple systems and aggregate the responses asynchronously at the presentation tier

Forward Service Cache

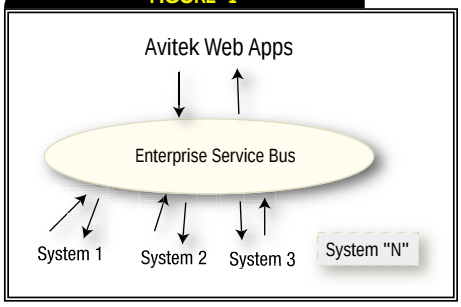
The "Forward Cache Service" use case addresses the problem of needing to expose data from a back-office system to the presentation tier. Although the presentation tier could easily interact with the back-office system through a request/response paradigm, there are several reasons why this could be impractical:

1. The back-office system is incapable of sustaining the load required to support the front-end presentation tier.
2. The latency for request/response would exceed the tolerances of the presentation tier.
3. There would be risk associated with exposing the back-office system directly to the presentation tier – stability, or impacts on existing service levels.
4. The back-office system could be in a different geography than the presentation tier; if links go down between the two data centers, the data should still be available to end users.

The ESB can be used to *reliably* forward changes to a cache in the presentation tier. The key word here is "reliably." In a distributed, SOA-based environment, careful attention must be paid to how systems interoperate, and what would happen in the event of failures or downtime. In many situations, systems are not capable of providing the necessary message resend and "in-doubt resolution" required for this type of reliability. An ESB can offload this complexity from the systems (see Figure 2).

The ESB, by its definition, is a distributed services network capable of reliable communications between any two entities. The deployment options provided by ESB providers will allow the quality of service to be tuned to the exact needs of the application. Based on industry-standard JMS, the two entities communicate reliably using standard interfaces; the ESB handles the

FIGURE 1



Modified Avitek architecture

The ESB: Your Web Services Network

At this point, you may be wondering "What about Web services"? The promise of Web services is certainly the ability to integrate across disparate technology, geographic, and organizational domains. Think of the ESB as a robust network for Web services. The ESB, built on the proven facilities of the Java Message Service (JMS), provides the key features needed to build a reliable, secure, manageable, and highly performing backbone for your interconnected services. In fact, the underlying backbone of an ESB has become a recognized SOAP transport in many of the available Web services toolkits. The ESB will provide interpretation and implementation for many of the emerging Web services standards, shielding portal developers from the details of network level interoperability between disparate services. Many of the complex Web services issues, such as routing and work flow, transactions, single sign-on security, auditing, advanced monitoring, and management, will be handled through the ESB infrastructure.

ESB Provider: Value-Added Services

As we mentioned earlier, many ESB vendors will offer value-added services to assist in creating a robust integration environment. In the context of the "Forward Cache" use case, here are some other things to consider:

- ***Connectivity:*** The "J" in JMS stands for "Java," so does that mean all systems must be Java-compliant? Certainly not! The ESB acts as the "Switzerland" of SOA environments, integrating disparate technologies into a common interoperable network. Look for ESB providers that support native C/C++ implementations, MS/.NET, native HTTP/SOAP support, and higher order adapters (JCA or custom) to automate the integration process.
- ***Transformations:*** The Avitek demo incorporates a transformation embedded at the presentation tier to transform different representations of medical records into the Avitek format. However, in a federated development environment, a developer with domain knowledge of the system may build the transform to a canonical XML format. In this case, wouldn't it be nice to relocate the transform close to the origination point (and the developer)? Look for ESB providers who allow configuration of transforms anywhere along the ESB.

complexity of routing and guaranteeing the delivery of the change notification.

Fortunately, the Avitek demo would not need to change drastically to take advantage of the ESB. Given the ESB's standards-based approach, the ESB provider hooks in through the JMS interface. Avitek's demo includes an MDB listening on a JMS connection to "upload" XML records into the MedRec database. Once the records are loaded into the database, they are available to the front-end portal using standard techniques to query the database. Of course, the pattern could also be enhanced to accommodate record deletion requests, or even partial record updates (see Figure 3).

But how does this differ from standard JMS providers? JMS providers provide asynchronous, guaranteed messaging capability in a single domain. Attempting to link multiple JMS messaging domains typically requires some type of custom bridge to forward messages reliably between multiple domains. However, ESB providers provide native, end-to-end JMS communications in a distributed and federated environment, obviating the need for custom bridges. Furthermore, the ESB provides additional standards-based connectivity, such as Web services and JCA adapters (see sidebar, "ESB Providers: Value-Added Services"), allowing the ability to flexibly deploy the services anywhere on the ESB.

One other consideration is how the data is cached in the presentation tier. The Avitek demo relies on persisting a standard XML Schema to a relational database. However, in a diverse services environment with many varying units of information, it might be preferable to store and process XML of different formats without needing to define rigid relational structures. ESB providers may offer an embedded XML database, offering a "schema-less" approach to storage, retrieval, and query of XML documents, drastically reducing database administration time to accommodate changes in service data on the back end.

There are some alternative approaches that we should discuss here for comparison and contrast to the ESB. BEA WebLogic 8.1 offers several different approaches to provide this type of loosely coupled, reliable messaging between disparate domains.

Reliable Web Services

BEA WebLogic 8.1 provides a new feature called "Reliable SOAP Messaging." This feature allows asynchronous, reliable messaging between two different WebLogic servers. Although SOAP/Web services is a standards-based approach, WebLogic 8.1 currently implements reliability through proprietary SOAP headers and interaction protocol. It is assumed that the feature will some day be implemented using a standards-based approach such as "WS-Reliability." Even with a standard HTTP-based approach such as "WS-Reliability," certain use cases that require high throughput and low latency will be better served through a true "end-to-end" JMS solution.

To reliably connect two systems across disparate networks and geographies, some type of infrastructure must be colocated in each domain to provide the necessary "store and forward" functionality in the event of failures. ESB infrastructure is typically "lighter-weight," easily managed, and solely focused on integration services. Furthermore, deploying ESB infrastructure into a .NET environment, or even an alternative J2EE infrastructure, might be more palatable to the receiving organization. Planning for an ESB up front could certainly avoid problems down the road.

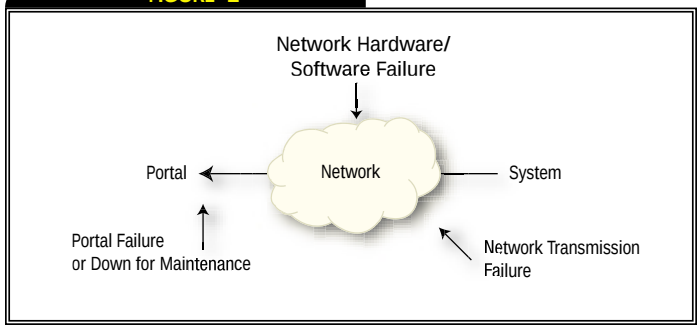
JMS Messaging Bridge

BEA WebLogic 8.1 offers a JMS messaging bridge that allows the ability to reliably forward messages between two different JMS implementations. While this certainly offers advantages over an HTTP/Web services approach, the infrastructure for the ESB provides the necessary forwarding, routing, and message optimization natively within the bus. The ESB also provides a cohesive management and security environment, simplifying deployment.

Federated Query

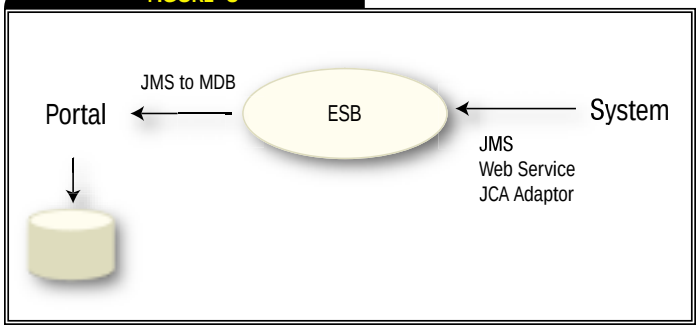
This "Federated Query" use case addresses the problem of needing to query multiple back-end systems from the presentation tier. Unlike the "Forward Service Cache" use case, the data in the back-end

FIGURE 2



Portal integration points of failure

FIGURE 3

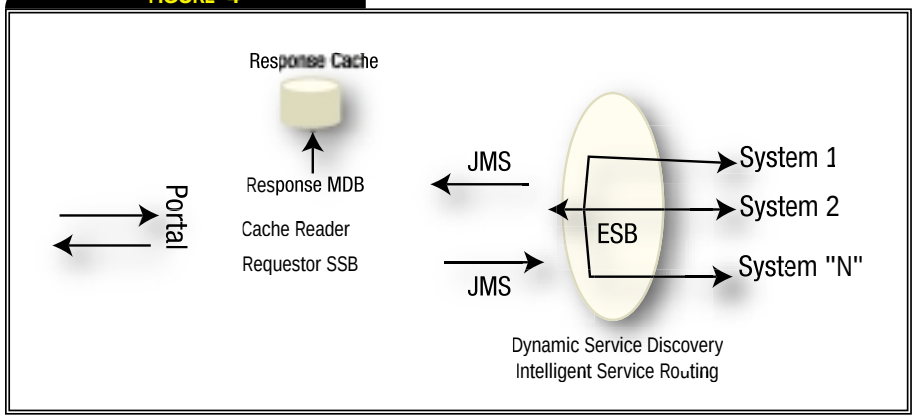


Reliable portal integration with the ESB

Confluent Software

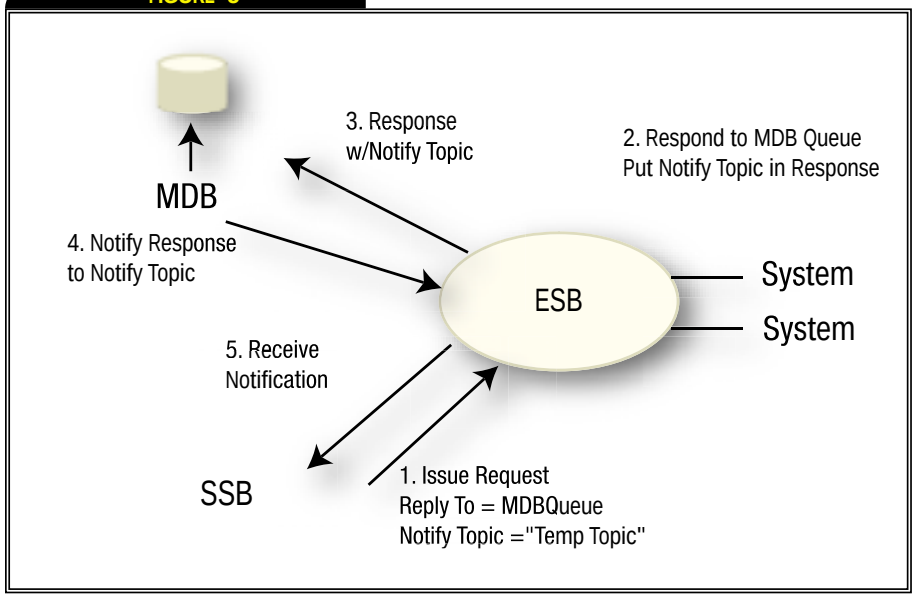
http://www.confluentsoftware.com

FIGURE 4



ESB communications interface

FIGURE 5



Combination Real-Time and Long Duration Request

system can't reasonably be cached. There could be several reasons for this:

1. ***The rate of data change in the back-end systems makes it unreasonable to cache the data:*** Querying the cached data could lead to data inconsistency and provide incorrect results.
2. ***The volume of data is too great:*** It is not technically or economically viable to cache the data forward.

Another aspect of the federated service query pattern is that requests may take too long to complete (in some cases, days!). For example, a single system may involve some type of manual intervention (such as an approval) to complete the workflow. The ESB provides a great foundation for this pattern due to its inherent asynchronous nature.

The Federated Query pattern comes in at least two variants. The ones we discuss here vary based on the duration of the query.

For the two varieties of Federated Query patterns discussed in Table 1, the ESB exploits JMS's underlying "Publish-Subscribe" messaging paradigm to efficiently fan out requests to multiple back-end systems (in this case, subscribers). While these patterns illustrate a very simple approach to implement the functionality, the ESB provides the core facilities to implement a wide range of techniques. The ESB provides a rich set of communication models that can accommodate a wide variety of interactions between the portal and back-end systems with utmost efficiency.

To illustrate this point, let's consider a query to three back-end services. If each service took three seconds to process, invoking these services sequentially would take a minimum of nine seconds. The ESB allows the execution of the services to proceed in parallel, making the overall service execution time equal to the longest-running service (in this case, three seconds). While this may be implemented using a centralized, multi threaded technique, the ESB allows the concurrent processing to be distributed across the bus, eliminating the centralized bottleneck and providing greater scalability potential.

Federated Query: Real-Time Request

For this case, we will enhance Avitek to provide the ability for a physician to query multiple back-end systems to determine the list of patients who have received blood within a certain time frame. To implement this feature, we're going to take advantage of WebLogic Server 8.1's new "JMS Wrapper" support. This feature provides the ability to efficiently send or receive JMS messages directly from an EJB or a servlet. WebLogic Server 8.1 will efficiently manage a JMS connection pool to ensure messages are routed quickly between the portal application and the JMS Provider (or in our case, the ESB!). Here's how it works:

A stateless session bean contains a method that implements the widely accepted JMS "ReplyTo" pattern. When the client (servlet, JSP, or other) invokes the method on the stateless session bean, the method publishes the request to the ESB, which in turn fans out the request to services listening on the configured topic. Examples of such services are JMS clients, Web services, or even JCA Adapters interacting with applications.

The stateless session bean defines a method to allow the client to send an arbitrary string request. This could be formatted in XML.

```
public boolean sendRequest(String requestData,
ArrayList a) {
```

Once the JMS objects have been established, the standard "JMSReplyTo" model is used to publish a request with the "return address". All responses will be returned to this instance of the stateless session bean (see Listing 1).

Finally, this example will wait for a specific number of responses from services on the ESB, or until a timeout occurs (see Listing 2).

Federated Query: Long Duration Request

For this case, we'll build on the previous example, but assume that responses from services on the bus will return in a much

TABLE 1

FEDERATED QUERY TYPE	SCENARIO DESCRIPTION
Real-Time Request	When a user requests a page from the portal, the portal sends a request to the ESB. The ESB routes the request to the appropriate systems on the bus, which respond in real-time to the original requestor. The portal application collects the responses (within a time period) and displays the results back to the user.
Long Duration Request	A user logs into the portal application and launches a query to the ESB. The ESB routes the request to the appropriate systems on the bus, and delivers the responses back to the same portal session. The user is free to use other features of the portal while the requests are accumulated.
Combination Real-Time and Long Duration	This pattern combines the previous two patterns. When the federated query is issued, the portal waits for a fixed amount of time and then returns the aggregated results to the user. Any "late" responses are asynchronously aggregated by the portal and available on subsequent requests by the user.

Federated Query pattern variants

Listing 1

```
// Create a temporary topic for services to reply to
Topic tempTopic = session.createTemporaryTopic();
TopicSubscriber subscriber = session.createSubscriber(tempTopic);
// Start the connection
connection.start();
// Set the temporary topic in the header
message.setJMSReplyTo(tempTopic);
message.setText(requestData);
publisher.publish(message);
```

Listing 2

```
// Wait for a specific number of replies, or until
// a timeout occurs
int num_replies_received = 0;
while (num_replies_received < m_num_replies) {
  Message m = subscriber.receive(m_receive_timeout);
  if (m == null) {
      // No message recieved
      break;
  }
  log("Received a message.");
  num_replies_received++;
  if (m instanceof TextMessage) {
       TextMessage tm = (TextMessage)m;
       a.add(tm.getText());
                 log("Added message to the list.");

  }
}
```

more unpredictable fashion. This pattern would allow the portal user to browse to other areas of the Web site, while responses asynchronously aggregate in the user's session. The user may even log out and log back in to check on the status of the request (see Figure 4).

For this case, a message-driven bean is used to asynchronously collect responses and persist them to a database. This is another case where a native XML database might come in handy. If systems respond with varying XML response information, it may be desirable to simply persist the entire XML response into the "schema-less" database.

One key characteristic of the ESB is that all services are tied together through a "loosely coupled" communications interface. One advantage relevant to this case is the ability for new systems to come online and be instantly included within the federated query. Since the underlying mechanism is "Publish-Subscribe," new services can simply subscribe to the relevant topics and receive query requests from the portal. An ESB permits dynamic discovery and intelligent, configuration-driven routing (content based, context based, etc.) at deployment time, allowing the portal to be shielded from service changes on the ESB.

Federated Query: Combination Real-Time and Long Duration Request

In some cases, it may be desirable to gather results from various systems on the bus within a certain time frame (real-time request), but continue to collect the responses asynchronously after the initial time frame has expired (Long Duration Request). To accomplish this, we introduce the notion of a separate "Notification Topic" to allow the stateless session bean (SSB) to receive notifications when the MDB has successfully persisted a response to the cache. The SSB can use arbitrary business logic to determine when to stop waiting, allowing the portal application to read from the cache and present the results to the user (see Figure 5).

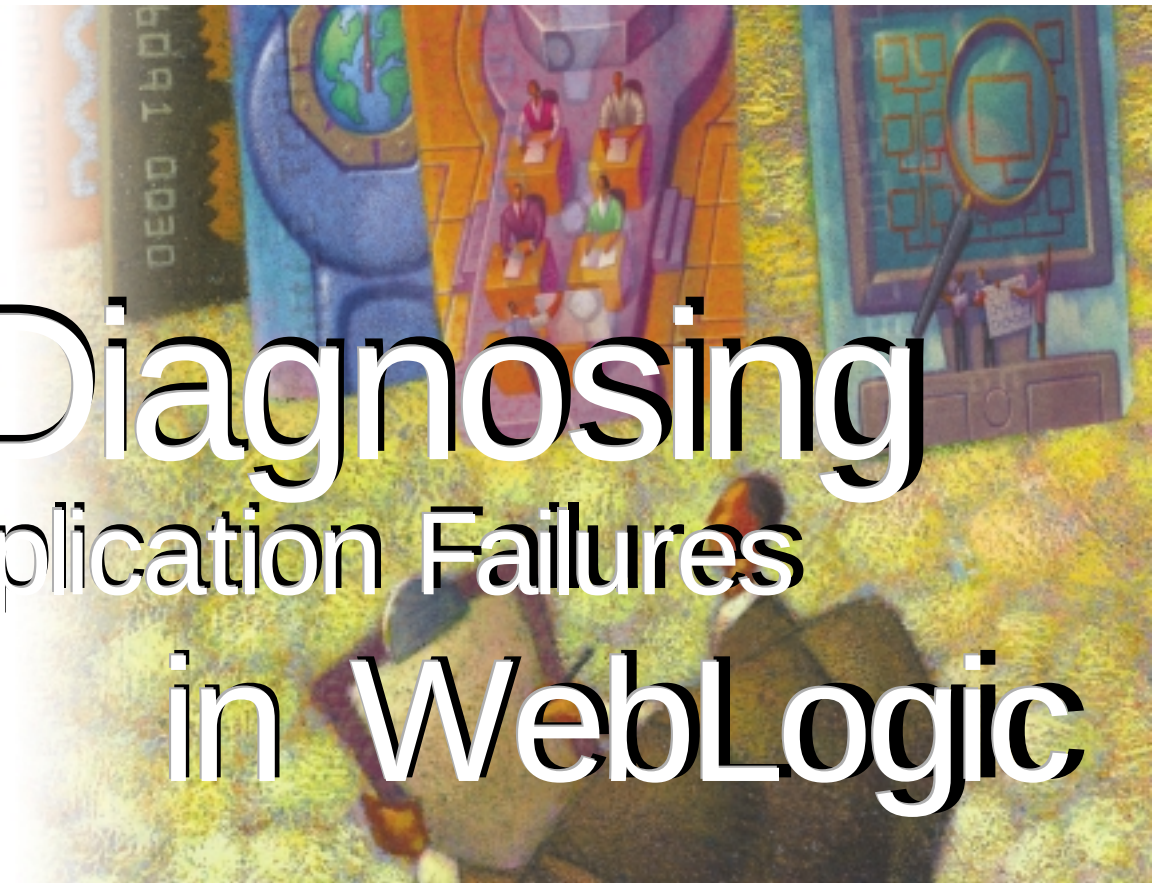
Summary

While BEA WebLogic Platform 8.1 is a robust platform capable of handling many application scenarios (including portals!), there are certain types of portal integration scenarios that require as the enterprise service bus. A fundamentally distributed network based purely on standards, the ESB will create highly agile, interoperable, and reliable service-oriented networks to link services together both within and beyond the enterprise. As Web service standards evolve to incorporate many of the semantics already available from JMS providers today, ESB providers will continue to implement standards and create interoperable networks that allow JMS apps to talk to Web services apps, and vice versa. In the meantime, an ESB will add significant value to portal and Web application developers attempting to solve complex integration scenarios involving heterogeneous systems, geographies, and organizations.

In our next article, we'll discuss how operational awareness, business activity monitoring, and integrated workflow can add additional value to these applications.

BEA Systems

http://dev2dev.bea.com/useworkshop



Diagnosing Application Failures in WebLogic

BY **VITALIY STULSKI**

AUTHOR BIO

Vitaliy Stulski is a Java developer and architect at TeaLeaf Technology, a provider of solutions to help businesses ensure the accuracy of their Web applications. He has extensive experience in the design and development of large, highly scalable, distributed enterprise applications based on the J2EE platform. Vitaliy also has several years of experience with BEA products, especially WebLogic Server.

CONTACT...

vstulski@tealeaf.com

Imagine. You're designing and developing a highly complex Web-based application. This app will serve thousands, or even millions, of customers. It will be deployed on hundreds of servers. Web and application servers will interact with a multitude of third-party services, no doubt accessing internal and legacy applications as well as issuing queries to a variety of databases. Furthermore, this system will require a great many tools and frameworks (from different software vendors, no less) that will perform different tasks and interoperate with each other in a highly dynamic manner.

In a standard development cycle you'll do your best to collect all known and necessary requirements, perform system analysis, design database schemas, and define integration points. Throughout this cycle, of course, you follow existing best practice recommendations. All of this is a prelude to your application design. This process can require thousands of person-hours, and will have to be tracked through thousands of milestones, big and small. You are confident that all of the difficult decisions have been anticipated and ameliorated, that the compromises made by the myriad independent groups within your company and your outside development partners have been managed. All of us are very familiar with this process, one that requires equal parts genius and patience.

The application is then deployed.

Problems occur.

After the initial panic subsides, you begin the triage process. Unfortunately, you don't always know why the application failed, for whom it failed, or the exact nature of the failure. Your application users may be based primarily in Japan, but your servers are hosted in Arizona, the support team is on the U.S. East Coast, and development is in California. The simple fact is that even identifying exactly what happened could take days or weeks, not to mention determining the implications of the problem to the application's users.

WebLogic is one of the best homes for complex J2EE applications on the market today. It makes application development, deployment, and management very convenient. However, the application server itself can address only very common problems during the application life cycle. The reality of today's complex applications typically requires better diagnostic capabilities, particularly in post-production.

Problem

After going through this scenario more than once – being frustrated by bugs, glitches, service

USING LOGGING FILTERS CAN

outages, and other sorts of problems – application architects will eventually decide to refocus their priorities to better anticipate production diagnostic. After all, the medical community has taught us that prevention is simpler than treatment. The same applies to software development. It is infinitely more efficient to prepare for problem resolution instead of waiting until problems occur. The question is, when entering the treatment phase, how can we avoid time-consuming application triage when that very application has already been shipped and is in use by customers?

This requires design of a diagnostic subsystem with the following characteristics:

- ***Descriptive and unified logging for any kind of messages:*** This logging should cover not only the applications themselves, but also all services and components the application might access. Recording of such messages should be based on a common logging framework, application code instrumentation, and binary code instrumentation, as well as possible third-party component log data normalization to a format that makes it possible to correlate the messages that arrive from the native logging framework;
- ***Good granularity of messages:*** When an error occurs it is very important to precisely identify the components that caused the problem, as well as the implications to the users affected by the service outage. It is very important to be able to troubleshoot each and every application module in great detail, despite the degree of complexity that is inherent in the app.
- ***Highly flexible logging configuration:*** Easy-to-use and descriptive views into the data collected by different management components. During the troubleshooting phase, we have to keep analyzed data to manageable amounts. It is all too common to be buried by volumes of data, but still lose track of the pertinent information. Usability is a crucial issue in every application logging strategy success. Nobody wants to use disorganized data, inconveniently collected. For all application support layers it is important to intelligently escalate from triage to problem resolution, filtering out nonrelevant information. Ultimately, it is vital to deliver the relevant information, in a manageable amount, so next-stage support can begin working toward problem resolution with no additional requests.
- ***Ability to support proactive problem identification and resolution scenarios:*** Through real-time alerts and notifications (e-mail, IM, etc.).
- ***Strong integration of logging data:*** With incident tracking system, including descriptive reports and views.
- ***Ability to support reactive problem mining:*** Running any business today requires compliance with high-quality standards that demand not only fixing a bug, but also determining the implications of such a bug. This includes uncovering each problem aspect, correlating the problem to all affected users, and providing necessary service to them. This goal is best addressed via highly tunable queries and data mining tools that will make all logging information collected by all your applications easily accessible.

Solution

The most common solution today to triage problem identification is that of generic logging. Typically, the application architect selects the logging strategy used by developers to issue logging messages to event recording subsystems directly from application code. It is possible to use publicly available logging frameworks, such as log4j or java.util.logging, or to develop one in-house. Regardless of the approach taken, it is necessary to unify this logging framework throughout all applications, as well as align output from different loggers into a common format. For example, using log4j it is easy to standardize logging infrastructure, create a loggers' hierarchy, extend the logging subsystem with appenders, and manage log sources. It is critical to have control over the output of each and every logger and to be able to feed them into a single searchable repository, along with preserving all-important structured data such as timestamps, server and application names, component names, and so on. Moreover, if the application uses third-party components and services it is good practice to make those logs searchable as well. In this way, it's possible to correlate problems the application reports with records in those logs.

The question is: Is it possible to come up with logging strategy and logging messages in the application code once and forever? A well-designed logging subsystem cannot address all possible failure points in a complex application without the risk of being overwhelmed with logging data, or overloading the application code with diagnostic instrumentation. Even prudently designed log levels would be insufficient in segregating valuable information related to the "noise"problem. Fortunately, it's possible to construct a dynamic logging solution using on-demand automatic code instrumentation. Currently, there is a wide variety of tools and frameworks available that allow different ways to instrument the code: from source code extensions by aspect-oriented pointcuts to bytecode patching using BCEL, or via other tools such as OC Systems' Aprobe. These technologies manage architect or developer concerns at almost every stage of the application life cycle by reducing logging data to desirable amounts. Using the tools mentioned above, or equivalent techniques, it's easy to develop a set of components (or probes) that will address different types of concerns within an application and apply them only when necessary and only to important application parts. In other words, instrumentation achieves necessary logging granularity on both component and code levels. At the same time, code complexity stays constant. Developers can, therefore, clearly separate diagnostic tasks from the actual business logic that makes those tasks loosely coupled within the application code.

The other side of the problem within the enterprise infrastructure (and probably the most mysterious one) that we need to address is that of application user behavior. Typically, use cases describe numerous scenarios of how applications are intended to be used, but there are always unanticipated use cases – a user that finds a way to bypass tested paths and is met with an application error. That's why it's very important to incorporate user session context during any logging task design and implementation. Modern application servers are well designed to isolate concurrent users from each other along with performed activities. WebLogic Server proved to be the fastest J2EE server on the market, which means it efficiently manages underlying hardware resources such as memory, threads, and sockets. Poorly written applications, however, will seriously impact even the best application server performance.

Starting with version 2.3, the servlet specification has a nice hint for a user context problem solution. In the chapter named "Filtering," we can get a good idea of logging and auditing filters. It's really up to architects, site administrators, and application support to determine how much infor-

mation about user interactions is necessarily logged. Servlet 2.3 filters allow you to dump certain request fields and parameters, along with response body, without modifying any of your servlets and JSPs. For example, TeaLeaf Technology's J2EE filter collects all information about request parameters, attributes, and other request-specific data along with the whole response body and writes it to the TeaLeaf RealiTea server for storing and post-processing. Along with data collecting, the filter creates unique context identification IDs that can be reused by downstream components for binding additional logging data. Unique IDs create a context that can be used for grouping all logging activity across all application components, which will allow the user to isolate the session that caused the problem and replay the steps that led to it.

Making All Logging Techniques Work Together

Servlet 2.3 API filter.

The Servlet 2.3 specification introduced filters as an integral part of the servlet container. While the specification itself states that filters are an ideal place for logging different sorts of user interaction, it is still difficult to find good examples of how to do this type of logging. Without diving into many details deserving of a separate article, our proposed logging framework is going to use Servlet 2.3 filters for the following purposes:

- To log request information
- To log response information
- To introduce a unique hit ID that will be passed to WebLogic internal tracing API

Each of these steps are fairly straightforward, based on collecting data by calling request object methods, wrapping response object methods and underlying output stream, generating globally unique hit IDs (based, perhaps, on unique hardware attributes where the program will be running and combining with the precise moment when the call occurred), and finally appending this ID to the running thread by calling the WebLogic tracing API.

```
weblogic.trace.Trace.beginTrace(uniqueId); //
uniqueId
```

Above is the byte array that is created for each hit.

WebLogic Tracing

Internal WebLogic tracing allows the user to propagate any byte array data throughout calling chains inside a single JVM or across different JVMs. As I stated earlier, I will use this feature to associate each particular Web site hit with all calls that will be executed by downstream components while generating an appropriate response. In other words, it is a matter of a single function call from any method in your application to reach information about the exact Web hit that caused this method to run.

```
byte uid[] =
weblogic.trace.Trace.currentTrace(); //retrieving
hit id recorded on previous step
```

Instrumentation

It's now possible to add dynamicity to our logging infrastructure by using tools like BCEL, OC Systems' Aprobe, or AspectJ to actually instrument particular parts of your application with logging messages. Logging messages are required to fetch the hit ID from the tracing context described in previous sections and to add as much information as needed. Tools such as Aprobe or AspectJ allow defining pointcuts where a developer wants to insert logging messages in a useful manner. For example, in Aprobe the callback class supports the published interface and a list of methods that will be instrumented. In AspectJ, it is pointcut definition and code that will be inserted to specified methods.

Glue

The final, and most important, component of the whole infrastructure is the system that will store all data that will be logged by all of the loggers and make it searchable. You can select different strategies, from storing data in text files and then importing them into a relational database, to logging directly to a database or using optimized logging solutions that enable storing and querying this data in real time (such as TeaLeaf RealiTea). "Glue server" is very important because tracing the context we inserted with the servlet filter, propagated by WebLogic tracing and used by logging code, will become relevant only in a central location where you can correlate data that came from your servlet hosted on Machine A to data from an EJB that is being hosted on Machine B.

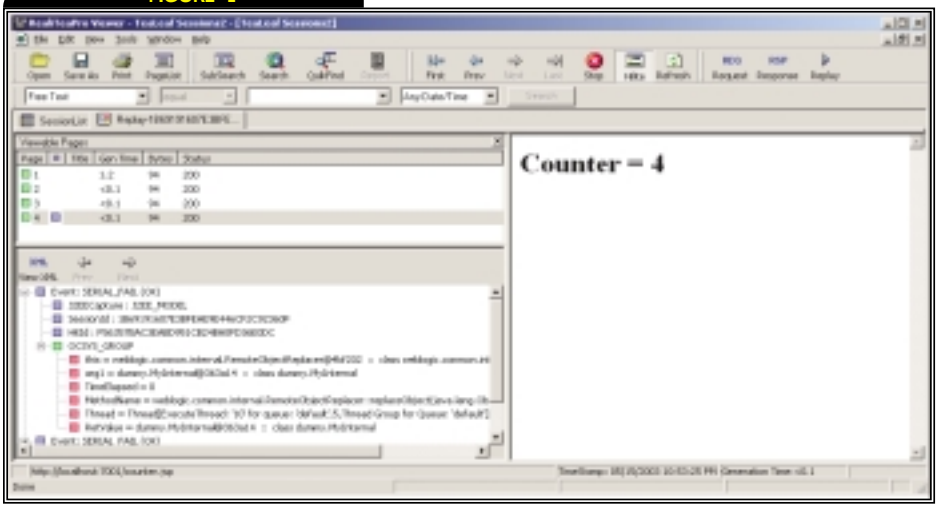
Real-World Scenario.

Let's imagine that our application is deployed to several cluster nodes. After running successfully for a period of time, we suddenly receive several serialization notifications from the replication subsystem.

Those messages contain information that some object cannot be serialized. Using standard logs provides little insight into the exact context of this problem. We don't know what servlet or JSP code attempted to insert invalid attributes into session context or, more important, exactly which object is not serializable. To simulate the problem described above we can use a simple chunk of code that simply counts visits to a page; inserts a serializable object into session context each time it is accessed, increasing the counter; and only on hit #3 will it insert an object that has a nonserializable member:

```
<%@ page import="dummy.*" %>
<%
Integer i =
(Integer)session.getAttribute("counter");
int iv = (i==null)?1:i.intValue()++;
session.setAttribute("counter", new Integer(iv));
if(iv==4) {
                    session.setAttribute("BADAT
                    TRIBUTE", new MyData());
```

FIGURE 1

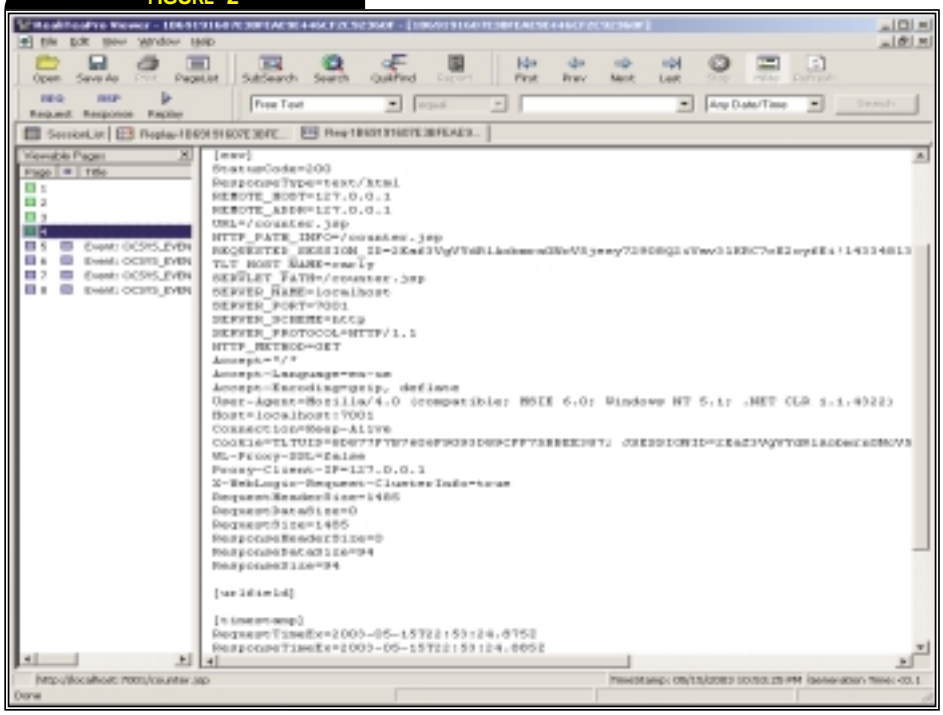


Session replay with erros loggeg at Hit #4

Hewlett-Packard

http://devresource.hp.com/wldj.html

FIGURE 2



Request parameters recording on Hit # 4

```
}
%>
<html><body><h1> Counter =
<%=iv%></h1></body><html>
```

Let's make our example a little more complex by making one of the MyData members (MyInternal) nonserializable:

```
package dummy;
import java.io.Serializable;
public class MyData implements Serializable {
          String str = "dummy string";
          MyInternal mi = new MyInternal();
}
And MyInternal source code:
package dummy;
public class MyInternal {
   Object o = new Object(); // this is
   non-serializable member
   String str = "string";
}
```

Real-world situations can easily be even more complicated – dynamic page code could be much more complex; session attribute insertion could be less straightforward; the application could use a controller that glues together different presentation, model, and data access components. All that could tremendously complicate problem diagnosis and resolution.

To solve this problem we can use TeaLeaf's filter component in conjunction with WebLogic's tracing capabilities and code instrumentation. The only steps required are to install the filter on our application (this step has to be performed only once per application), make WLS run using tracing (Dweblogic.TracingEnabled=true), and configure the instrumentation patch. The filter in this scenario is performing the following tasks:

- HTTP traffic logging (request and response data, including HTML/XML, served back to the client);
- Creating and binding tracing context for application servlet and downstream components that will be called from the servlet;
- Feeding all data into TeaLeaf RealiTea Server, allowing the user session to be replayed and making the entire session searchable and accessible in realtime. Alternatively, the RealiTea Server could record all of the data into log file format locally on the server.

WebLogic tracing provides logging code and instrumentation patches with unique context identifiers that can be accessed from any downstream component and that can propagate the context across JVM boundaries in case of RMI calls.

For code patching in this example it is possible to use OCSystems' Aprobe, BCEL, or even AspectJ. The main idea is to add additional verification to the method that verifies each and every object before writing it to a stream (weblogic.common.internal.RemoteObjectReplacer.replaceObject) if it is serializable, and proceeding with event issuing if it is not:

```
…
if(returnValue instanceof Serializable)
return;
   …
```

In case of verification, it's necessary to extract the current call trace context using the WebLogic tracing API and join a logging message to the request/response data logged by the filter:

```
byte b[] = weblogic.trace.Trace.currentTrace();
// get logging context
if(b!=null && b.length==64) {
   sid = new String(b,0,32);  // get unique ses-
sion id
hid = new String(b,32,32); // get unique hit id
// record an event with your favorite logging
API
logger().error(sid, hid, "SERIAL_FAIL",
SymbolTable.getPrintableMethodName(super.methodId
));
}
…
```

Finally, we have to instrument the application and let it run. After executing a four-hit session we can find the following information logged (in the example I'm using RealiTea Viewer, TeaLeaf Event API and capture filter; see Figure 1).

Now we find that the problem is happening exactly on hit number 4 of this session. We also clearly see that the object that cannot be serialized is an instance of MyInternal class, and that all request and response data sent to the WebLogic Server by the browser and generated by the servlet is captured automatically via the filter. In short, we can now easily find each and every user affected by the same problem after the fact (see Figure 2).

Conclusion

Having logging filters installed and running on your production servers and your centralized logging system, combined with custom logging instrumentation, will significantly reduce the time required to diagnose serious application problems. This process is further improved by constantly monitoring complex Web applications through the initiation of proactive alerts and notifications. On the other hand, a full record of user interactions will help address not only technical problems, but also the business issues and implications of how application failure affects all site users.

www.LinuxWorld.com

The Leading Magazine for Enterprise and IT Management

LinuxWorld Magazine

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *LinuxWorld Magazine* is aimed squarely at providing this group with the knowledge and background necessary to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *LinuxWorld Magazine* does not feature low-level code snippets but focuses instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month presents a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features include:

- *Advice on Linux Infrastructure*
- *Detailed Software Reviews*
- *Migration Advice*
- *Hardware Advice*
- *CEO Guest Editorials*
- *Recruiting/Certification Advice*
- *Latest News That Matters*
- *Case Studies*

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE $71.76

YOU PAY ONLY $49.99

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR VISIT WWW.SYS-CON.COM

The World's Leading i-Technology Publisher

This month I was again inspired by the weblogic.developer.interest.transaction newsgroup on newsgroups.bea.com – if you weren't listening last time I plugged this newsgroup, listen now; one day, it might save your life!

Transaction Not Supported? Just Say No!

ACT NOW TO AVOID GRIEF LATER

BY PETER HOLDITCH

Be Paranoid, Be Very Afraid

A quick guess at what "NotSupported" does might lead you to wonder if it's just a repeat of Never and worry about the redundancy, whilst musing on "Supported" might make you paranoid about what is really going to happen at runtime. Well, be very afraid of these two innocent-sounding attributes!

As you know if you've used EJB container-managed transactions, you have several options available in declaring your methods' transactional behavior, namely NotSupported, Required, Supports, RequiresNew, Mandatory, or Never. I guess the one with the clearest intent is "RequiresNew" – the method requires a new transaction context, so the container will create one before running the code. Next, "Required" – a transaction is required to run this method, so if the caller didn't supply one, the container will create a new transaction context for the method to run in. In "Mandatory" – the method requires a transaction context and it's an error by the caller to not supply a transaction context prior to the call. Then "Never" – a transaction context must never exist when this method is called, and if one does it's an error. There are still two more options, Supports and NotSupported; these are much less imperative sounding than the other choices and, as always in software design, precision is everything!

"*The Supports transaction attribute must be used with caution. This is because of the different transactional semantics provided by the two possible modes of execution. Only the enterprise beans that will execute correctly in both modes should use the Supports transaction attribute.*" (Section 16.7.2.3)

This is clearly bad news, and anyone with any sense will take this as a cue to give Supports a wide berth unless there is a very good justification for not doing so.

However, the spec gives no such warning about the NotSupported behavior, which I think is rather odd since it is even more dangerous.

The spec argues that the danger with Supports is that it makes your methods behave in different ways, depending on the calling context. This obviously is a bad thing, encouraging close (and hard to spot) coupling between the caller and callee and generally breaking the laws of good practice. At least Supports always behaves in the same way. What's not so clear from this section of the spec is in what consistent way this will behave. It simply makes the innocent-sounding statement that "The Container invokes an enterprise bean method whose transaction attribute is set to NotSupported with an unspecified transaction context." (16.7.2.1)

We Need Scooby Doo!

To get to the bottom of this mystery, we need a detective. After staking out the spec for a while from behind a bush we notice a light on after hours in section 16.7.5. In fact, this whole section is a horror story about the kind of things containers get up to in "unspecified transaction contexts" – I'll not quote it verbatim here, my fingers daren't type that kind of horror, but suffice it to say, containers can do whatever they like – start a new transaction per call, start a new transaction and lump multiple calls together in it, run with no transaction context at all. This all adds up to bad news. Your bean's behavior will change from application server to application server (and could even legitimately change between releases of application server) all this adds up to… zoinks! Portability problems. Which was one of the things you set out to mitigate by using J2EE in the first place. As a footnote, WebLogic's behavior in this situation (in the current release!) is to execute methods on Entity beans in their own fresh transaction context (equivalent to the RequiresNew behavior) and to execute methods on session beans with no active transaction (equivalent to a bean managed bean that hasn't started a transaction).

So, we have unmasked the felon – the innocent-looking unspecified transaction context! Remember kids, the moral of the story is: Don't use NotSupported, Never, or Supports for your container managed transactions, or you'll store up a whole load of grief for yourself (or for whoever inherits your code to maintain!).

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

DELIVERING .NET, JAVA, MAC OS X, AND XML TECHNOLOGIES

International Web Services Conference & Expo

Web Services Edge

SEPT. 30 — OCT. 2, 2003

3rd Annual

web services conference EDGE &expo

- *Take in tutorials covering .NET & Web services*
- *Listen to success stories*
- *Evaluate case studies & best practices*
- *Experience hands-on labs*

REGISTER TODAY!
CALL 201-802-3058
www.sys-con.com/edge
Register by September 5th and
SAVE Up To $200

2003 WEST

Santa Clara, California

KEYNOTE SPEAKERS

Vermeulen
CTO
amazon.com

Magee
VP, Oracle 9i
ORACLE

Litwack
Senior VP
Novell

Schmidt
VP, Systems Integration
BEST BUY

Event Sponsors: ORACLE Novell Education Sponsors: Microsoft Sun microsystems

Media Sponsors: WebServices Journal, JAVA Developer's Journal, WebSphere Developer's Journal, XML Journal, .NET Journal, WebLogic Developer's Journal, LinuxWorld Magazine, wireless Business Technology, LINUX Business Week, CF Advisor, ColdFusion Developer's Journal, PowerBuilder Developer's Journal, market WIRE, HSP Street.com, OASIS, SD Times, LinuxWorld.com, SAMS, Industry

Owned and Produced by: SYS-CON EVENTS

SPECIAL INSERT:

Web Services Edge West Conference & Expo Sept. 30 - Oct. 2, 2003

SPECIAL INSERT:
Web Services Edge West Conference & Expo Sept. 30 - Oct. 2, 2003

web services conference EDGE &expo

Conference & Expo

THE LEADING EVENT

for i-Technology Professionals

SEPT. 30 – OCT. 2, 2003

Santa Clara Convention Center

FEATURES & ATTRACTIONS

- 3 Days Packed with Education and Training
- Keynotes & Panel Discussions from Industry Leaders
- 60 Hard-hitting and Informative Seminars
- **FREE** Web Services Workshop Presented by Oracle
- **FREE** .NET Tutorial with Microsoft's Russ' Tool Shed
- Java University Certification Training
- Industry-Leading Certification Programs
- **FREE** IBM Web Services Tutorial
- "Birds of a Feather" Discussions
- Round Table Discussions
- Opening Day Welcome Reception
- SAMS Meet the Authors Hot Topics Lounge
- Compelling Case Studies & Best Practices
- Hands-On Labs
- Featured Product Demonstrations
- Exhibit Floor featuring more than 40 companies and hundreds of products
- Real-time SYS-CON Radio Interviews

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

WHO SHOULD ATTEND

- CEO
- CTO
- CIO
- IT Director
- Project Manager
- ▸ Software Developer
- ▸ Software Engineer
- ▸ Development Manager
- ▸ Application Developer
- ▸ Technical Director
- ▸ Analyst/Programmer
- ▸ IT Manager
- ▸ Technical Architect
- ▸ Team Leader
- ▸ Software Consultant

KEYNOTES & HIGHLIGHTED SPEAKERS

Allan Vermeulen

CTO, Amazon.com

Sept. 30 10:00 a.m.

"Web Services Foundations"

Allan Vermeulen, CTO and vice president at Amazon.com, directly oversees the Platform Technologies group. This group is responsible for guiding Amazon.com's technology architecture, including building and acquiring foundational components. Prior to his move to Amazon.com, Vermeulen was CTO and vice president of development at Rogue Wave Software. He holds a PhD in systems design engineering from the University of Waterloo.

John Magee

Vice President, Oracle9i Application Server, Oracle

Oct. 1 10:00 a.m.

"J2EE Development on the Grid"

John Magee is vice president of Oracle9i Application Server and Oracle9i Developer Suite at Oracle. Mr. Magee has over 14 years of experience in the enterprise software industry and has held positions in product development, product management, and product marketing. In his current role, he manages technical product marketing for Oracle's application server and development tools products, and is responsible for evangelizing Oracle technology initiatives around J2EE, XML, and Web services.

David Litwack

Senior Vice President, Web Application Development Products, Novell

Sept. 30 2:00 p.m.

"Business Integration and IT" Keynote Panel

David A. Litwack is senior vice president of Web Application Development Products, responsible for the development and advancement of Novell's secure Web services strategy. Mr. Litwack assumed his current position in July 2002 following Novell's acquisition of SilverStream Software, a company for which Litwack had served as president and CEO since 1997.

John Schmidt

Leader of Systems Integration and Middleware, Best Buy Co.

Sept. 30 2:00 p.m.

"Business Integration and IT" Keynote Panel

John Schmidt is the chairman of the Methodology Committee for the EAI Industry Consortium and leader of systems integration and middleware at Best Buy Co., a leading specialty retailer of consumer electronics, personal computers, entertainment software, and appliances.

Jon Bosak

Distinguished Engineer, Sun Microsystems

Jon Bosak organized and led the W3C working group that created the XML specification and then served for two years as chair of the W3C XML Coordination Group. At Sun, where he holds the title of Distinguished Engineer, Mr. Bosak sponsors projects intended to advance XML technology. He is currently chair of the Universal Business Language (UBL) Technical Committee of OASIS.

Dave Chappell

VP, Chief Technology Evangelist, Sonic Software

Dave Chappell is the vice president and chief technology evangelist for Sonic Software. He has more than 18 years of industry experience building software tools and infrastructure for application developers, spanning all aspects of R&D, sales, marketing, and support services. Dave has also been published in numerous technical journals, and is currently writing a series of contributed articles for *Java Developer's Journal*.

Anne Thomas Manes

Research Director, Burton Group

Anne Thomas Manes is a research director at Burton Group, a research, consulting, and advisory firm. Anne leads research for the Application Platform Strategies service. Named one of NetworkWorld's "50 Most Powerful People in Networking" in 2002, and one of Enterprise Systems Journal's "Power 100 IT Leaders" in 2001, Anne is a renowned technologist in the Web services space. Anne participates in standards development at W3C and OASIS.

Marc Fleury

President, JBoss

Marc Fleury, PhD, is chief technical officer for Telkel, Inc. He is the leader of the JBoss project (www.jboss.org), which is an open source EJB server. Marc is based out of Silicon Valley and founded the project upon leaving Sun Microsystems. He was one of the main developers behind JBoss 1.0 and 2.0. Marc is the "keeper" of the project. He founded the Jboss Group, a company regrouping the elite developers of Jboss to consult around Jboss.

Hotel & Travel

Reserve Your Hotel Room Now
At The Westin Santa Clara!
The Official Conference Hotel of Web Services Edge West 2003

The Westin Santa Clara
5101 Great America Parkway
Santa Clara, CA 95054

Arrangements have been made with the Westin Santa Clara, which is conveniently located at The Santa Clara Convention Center. Specially reduced rates have been secured at this luxury, full-service hotel.

Single Occupancy Room: $165.00
Double Occupancy Room: $165.00

All rooms are quoted exclusive of applicable state and local taxes which are currently 9.5% as well as the California State Tourism Tax of 0.045%. The above rates are group rates and are available for Web Services Edge 2003 delegates, over the show dates of September 28 – October 3, 2003, only.

To learn more about The Westin Santa Clara you can contact the hotel directly or you can make your reservations by calling Expo Travel International at (800) 829-2281 or (201) 444-0060 (direct). Fax reservations to (201) 444-0062. Credit card information is required to guarantee reservations and expedite confirmation. Confirmations will be mailed directly from the hotel, time permitting. All changes and cancellations should be made directly through Expo Travel International.

To make online reservations:
www.expotravel.com by September 12, 2003.

Reservations received after September 12, 2003, will be accepted on a space-available basis only, at the special rate, if available.

Contact Information:
The Westin Santa Clara Reservations:
Tel: 408 986-0700
Fax: 408 980-3990

Hotel Arrangements Are Easier Than Ever! You have your choice – contact the hotel directly or call us. The Official Conference travel agent, Expo Travel International.

Official Conference Travel Agent:
Expo Travel International
Toll Free: (800) 829-2281
Tel: (201) 444-0060
Fax reservations to (201) 444-0062

Driving Directions to Westin Santa Clara from San Jose Airport:
Highway 101 North. Exit at Great America Parkway/Bowers. Turn Right onto Great America Parkway; hotel is about 1.5 miles down on the right side.

SPECIAL DISCOUNTS AVAILABLE

Take advantage of the Early Bird and Preregistration values available right now, or save even more with a group of 5 or more. For special group discounts contact Michael Lynch at mike@sys-con.com, or by phone at (201) 802-3058.

PRODUCED BY SYS-CON EVENTS

SPECIAL INSERT:
Web Services Edge West Conference & Expo Sept. 30 - Oct. 2, 2003

SPECIAL INSERT:
Web Services Edge West Conference & Expo Sept. 30 - Oct. 2, 2003

Conference at-a-Glance

		JAVA	.NET	WEB SERVICES
TUESDAY, SEPTEMBER 30	8:00AM – 4:00PM	**REGISTRATION**		
	9:00AM – 9:50AM	The Next Phase in Evolution of J2EE	Using WSE 2.0	Web Services Management
	10:00AM – 10:50AM	Keynote - "Web Services Foundations" - Allen Vermeulen, CTO and Vice President, Amazon.com		
	11:00AM – 6:00PM	**EXPO OPEN**		
	2:00PM – 2:50PM	Keynote Panel Discussion - Business Integration and *i*-Technology		
	3:00PM – 3:50PM	Ant Applied in "Real World" Web Services	Smart Devices in the Enterprise	Building Interoperable Web Services Using WS-I Basic Profile
	4:00PM – 4:50PM	Developing Applications with SWT	Using the Mobile Internet Toolkit	Web Services Orchestration
	5:00PM	**OPENING NIGHT RECEPTION**		
WEDNESDAY, OCTOBER 1	8:00AM – 4:00PM	**REGISTRATION**		
	9:00AM – 9:50AM	Empowering Java and RSS for Blogging	Introduction to ROTOR	ID, Please. The Case for Giving Web Services an Identity
	10:00AM – 10:50AM	Morning Keynote - "J2EE Development on the Grid" - John Magee, Vice President, Oracle9i, Oracle		
	11:00AM – 4:00PM	**EXPO OPEN**		
	2:00PM – 2:50PM	Keynote Panel Discussion - Interoperability: Is Web Services Delivering?		
	3:00PM – 3:50PM	JUnit: Testing Your Java with JUnit	Using Portable .NET	WS-BPEL
	4:00PM – 4:50PM	JDK1.5: The Tiger	ASP.NET with Mono	UDDI: Dead or Alive?
	5:00PM – 6:00PM	Squeezing Java	Using WSE with IBM's Web Services Tool Kit	Web Services Choreography, Management, and Security - Can They Dance Together?
THURSDAY, OCTOBER 2	8:00AM – 4:00PM	**REGISTRATION**		
	9:00AM – 9:50AM	Leveraging AOP in JBoss	Success Story: Eiffel, .NET, and Design by Contract for the Financial Industry	Strategies for Securing Web Services
	10:00AM – 10:50AM	Technical Keynote		
	11:00AM – 11:50AM	Apache Axis	.NET IDE's	Web Services Progress Report
	12:00PM	**LUNCH**		
	1:00PM – 1:50PM	Meeting the Challenges of J2ME Development	Windows SharePoint Services	The Seven Habits of Highly Effective Enterprise Service Buses (ESBs)
	2:00PM – 2:50PM	Keynote Panel Discussion - Summit on Web Services Standards		
	3:00PM – 3:50PM	Simplifying J2EE Applications	BizTalk 2004	See www.sys-con.com for more information
	4:00PM – 5:00PM	Integrating Java + .NET	See www.sys-con.com for more information	See www.sys-con.com for more information

XML	MAC OS X
Introduction to Xforms	Introducing OS X (Panther) What's New?
Securing Your XML and Web Services Infrastructure	Programming Rich User Interfaces Using Cocoa
UBL - The Universal Business Language	Quick Applications Using AppleScript
Standards-Based Enterprise Middleware Using XML/Web Services	Java and OS X: A Perfect Marriage
XML and Enterprise Architecture: Technology Trends	Enterprise Java and OS X
Using XML Schemas Effectively in WSDL Design	Developing Web Services Using WebObjects
Canonical Documents for Your Business: Design Strategies	Cocoa, Carbon, Java: Application Frameworks for OS X (When to Use What)
XML and the Fortune 500	Securing OS X Applications
XML at Work in 'Fortune 500' Companies	Xserve: Ease of OS X and Power of Unix
XML Schema Best Practices	OS X for the Unix Developer
See www.sys-con.com for more information	Introducing Quartz: 2D Graphics for Apple
See www.sys-con.com for more information	See www.sys-con.com for more information

PROGRAM SUBJECT TO CHANGE

FREE Web Services Workshop
presented by

ORACLE®

October 1, 2003

SHARPEN YOUR SKILLS, DEVELOP YOUR CAREER

Web services? You've read all the ins and outs about it. You think you have the concepts pretty well figured out. Now you are not sure where and how to start developing your first Web service. Get the answer at this free Web services workshop offered by Oracle as part of its Oracle Developer Days roadshow!

Oracle's workshop is specifically designed to get you started with your first Web service project, with a combination of presentations and hands-on labs that take you deep into the technology and let you put in action what you've learned. Oracle's experts will be available throughout the workshop to answer all your questions and assist you while you are going through the labs.

The workshop gives tips and techniques on how best to develop and deploy Web services and addresses topics such as RPC and Document Style Web services, static and dynamic invocation, stateless Web services and more. The second part of the workshop is dedicated to the new J2EE API for Web services available as part of J2EE 1.4.

Going through the hands-on labs at your own pace, you will learn how to publish a Java class as a J2EE stateless or stateful Web service, publish a session EJB as a J2EE Web service, and publish a J2EE Web service using JAX-RPC.

Space is LIMITED to the first 100 attendees. Register now for this FREE workshop. Computers will be provided by the Oracle Developer Days team with all the necessary software, so there's no need to bring your own computer.

AGENDA

7:30-8:00 am – Registration
8:00-9:00 am – Session #1 – Best Practices for Web Services Development & Deployment
9:00-10:00 am – Lab #1 – Publish a Java Class as a J2EE Stateless or Stateful Web Service
10:00-10:50 am – John Magee, VP, Oracle – Keynote (BREAK)
11:00 am-12:00 pm – Expo Floor Time
12:00-1:00 pm – Session #2 (WORKING LUNCH) – J2EE APIs for Web Services
1:00-2:00 pm – Lab #2- Publish a Session EJB as a J2EE Web Service
2:00-2:30 pm – Expo Floor Time (BREAK)
2:30-3:00 pm – Lab #3- Publish a J2EE Web Service Using JAX-RPC

PRESENTERS

Arun Srinivasan, Director of Product Management, Java Tools, Oracle
Rob Clark, Director of Product Management, J2EE, Oracle
Mike Lehmann, Product Manager, Web Services, Oracle9*i*AS and Oracle9*i* JDeveloper, Oracle

FREE Oracle Tutorial when you register for a VIP Pass
Guarantee your seat when you register for a Full Conference Pass

Register Online at
www.sys-con.com/edge

SPECIAL INSERT:
Web Services Edge West Conference & Expo Sept. 30 - Oct. 2, 2003

REGISTRATION FORM

CONFERENCE: Sept. 30 – Oct. 2, 2003 **EXPO: Sept. 30 – Oct. 1, 2003**

Santa Clara Convention Center • Santa Clara, CA

THREE WAYS TO REGISTER FOR CONFERENCE

1) On the Web: Credit Cards or "Bill Me." Please make checks payable to SYS-CON Events.
2) By Fax: Credit Cards or "Bill Me" 201-782-9651
3) By Mail: 135 Chestnut Ridge Road, Montvale, New Jersey 07645, Attention: Registration

Please note: Registrations are not confirmed until payment is received.

Please complete sections 1, 2, 3 and 4

1 YOUR INFORMATION (Please Print) ☐ Mr. ☐ Ms.

First Name _____ Last Name _____

Title _____

Company _____

Street _____

Mail Stop _____

City _____

State _____ Zip _____ Country _____

Phone _____

Fax _____ E-Mail _____

2 PAYMENT METHOD: (Payment in full due with registration)

☐ Check or Money Order Enclosed (Registration confirmed upon receipt of payment)

Check # _____ Amount of Check $ _____

Charge my ☐ Visa ☐ MasterCard ☐ American Express ☐ Discover

Name on card _____

Card # _____ Exp. Date _____

Signature _____

Billing Address (if different from mailing address)

_____

3 PLEASE INDICATE YOUR CONFERENCE CHOICE

Total Registration fee $_____

	By 9/5/03	Before 9/26/03	Onsite
❑ **GP Gold Passport** Good for all three days of the .NET, Web Services, XML, Java, and Mac OS X Tracks, including preferred seating for the Oracle, IBM and Microsoft Russ' Toolshed Tutorials, Keynotes, Panel Discussions, and your choice of One Sun Microsystems Java University℠ Class Select one: ❑ Architecting Web Services Using J2EE (Oct. 1) ❑ Java 2 Platform: Architect Certification Fast Path (Oct. 2)	$1,295.00	$1,395.00	$1,495.00
❑ **3D Three Day Conference** (Does not include Sun Java Education)	$1,195.00	$1,295.00	$1,395.00
❑ **2D Two Day Conference** (Does not include Sun Java Education) (select any two days: ❑ Tue. ❑ Wed. ❑ Thurs.)	$1095.00	$1,195.00	$1,295.00
❑ **1D One Day Conference** (Does not include Sun Java™ Education) (select any one day: ❑ Tue. ❑ Wed. ❑ Thurs.)	$595.00	$595.00	$695.00
❑ **JU1 Sun Java™ University Class** Select one: ❑ Architecting Web Services Using J2EE (Oct. 1) ❑ Java 2 Platform: Architect Certification Fast Path (Oct. 2)	$695.00	$695.00	$795.00
❑ **JU2 Sun Java University Class** Attend both Architecting Web Services Using J2EE (Oct. 1) and Java 2 Platform: Architect Certification Fast Path (Oct. 2)	$1,195.00	$1,295.00	$1,395.00
❑ **VIP PASS** Good for access to the Exhibit Floor, Keynotes and Panel Discussions, Product Demonstrations, and your choice of (Select one): ❑ Microsoft Russ' Tool Shed (Sept. 30) ❑ How to Develop, Deploy, and Manage Web Services Using IBM Tools (Sept. 30) ❑ Web Services Workshop presented by Oracle (Oct. 1)	FREE	FREE	$50.00
❑ **EO Expo Only**	FREE	FREE	$50.00

4

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 Programmer
❑ 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 ❑	01 ❑
5,000 – 9,999	02 ❑	02 ❑
1,000 – 4,999	03 ❑	03 ❑
500 – 999	04 ❑	04 ❑
100 – 499	05 ❑	05 ❑
100 or less	06 ❑	06 ❑

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

❑ $10 million or more
❑ $1 million – $9.9 million
❑ $500,000 – $999,999
❑ $100,000 – $499,999
❑ $10,000 – $99,999
❑ Less than $10,000
❑ Don't know

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

❑ $10 billion or more
❑ $1 billion – $9.9 billion
❑ $100 million – $999 million
❑ $10 million – $99.9 million
❑ $1 million – $9.9 million
❑ Less than $1 million
❑ 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?

❑ Application Servers
❑ Web Servers
❑ Server Side Hardware
❑ Client Side Hardware
❑ Wireless Device Hardware
❑ Databases
❑ Java IDEs
❑ Class Libraries
❑ Software Testing Tools
❑ Web Testing Tools
❑ Modeling Tools
❑ Team Development Tools
❑ Installation Tools
❑ Frameworks
❑ Database Access Tools / JDBC Devices
❑ Application Integration Tools
❑ Enterprise Development Tool Suites
❑ Messaging Tools
❑ Reporting Tools
❑ Debugging Tools
❑ Virtual Machines
❑ Wireless Development Tools
❑ XML Tools
❑ Web Services Development Toolkits
❑ Professional Training Services
❑ Other [Please Specify]

If you require special assistance covered under the Americans with Disabilities Act, please call 201-802-3058 by September 16, 2003.

CANCELLATIONS, SUBSTITUTIONS, REFUNDS

Fax written request to SYS-CON Registration 201-782-9651. Requests for refunds received prior to August 29, 2003, will be honored, less a 10% handling charge; requests received after August 29, 2003, and before September 12, 2003, will be honored less a 20% handling charge. No requests for refunds will be honored after September 12, 2003. Requests for substitutions must be made in writing prior to September 26, 2003. No one under 18 is permitted to attend. No warranties are made regarding the content of sessions or materials.

Speakers, sessions, and schedule are subject to change without prior notice.

No solicitation by anyone other than official exhibitors, sponsors or marketing partners is permitted. Such behavior is cause for expulsion without refund.

What are ACID transactions? How do they change to work with Web services? And how do the ACID guarantees work when you must use compensating actions?

Transactional Web Services

COMPENSATION AND COOPERATION

BY **WILLIAM COX**

AUTHOR BIO

William Cox is a technical director in BEA's Office of the CTO, concentrating on transactional and portal architecture. He is a coauthor of BTP, the BTP Primer, and of WS-Transaction. William holds an M.S. and Ph.D. in computer sciences from the University of Wisconsin–Madison.

CONTACT...

william.cox@bea.com

Transactions – Formal and Informal

To many people, a "transaction" is a business exchange where money is traded for goods. To software engineers the meaning is more technical. Informally, a transaction implies that a group of activities is completed as a unit, so they all succeed or all fail together. This "all or none" semantic is fundamental to database access.

In the formal model, the groupings we call transactions have properties known by the acronym ACID – they are Atomic, Consistent, Isolated, and Durable (see Table 1).

Enterprise systems, including many parts of Java 2, Enterprise Edition (J2EE), support transactional semantics. For example, Enterprise JavaBeans (EJBs; for storage operations) and the Java Message Service (for messages) have transactional guarantees on their behavior. JTA/JTS (the Java Transaction API and the Java Transaction Service) provide standard APIs for accessing the transactional capabilities provided in compliant software.

ACID and Compensating Transactions

ACID transactions are critical to business interactions – a bank wants to ensure that your account is credited exactly once for a deposit, and you want to ensure that your account is debited exactly once for a withdrawal. Likewise, changes in the data stored in an EJB must be accessed only when it is internally consistent – transactions guarantee that consistent view.

To ensure consistency, typically all database entries being used by an ACID transaction are locked for the duration. If a transaction fails, the database state is rolled back to its previous state. This capability is provided by database vendors.

But locking cannot work across enterprises. When you make a hotel reservation, your travel agent cannot lock the hotel's reservation database for as along as the reservation exists (or even as long as a phone call) or the system would grind to a halt. Instead, an ACID transaction local to the hotel chain's database as a single unit of work (1) updates the room inventory, (2) adds information to the reservation table, and (3) generates a confirmation number. If the travel agent needs to undo that reservation, a compensating action is taken. Sometimes there's a cost to compensate – for example, if the reservation is cancelled too late, there may be a charge.

Compensation is specific to the way business data is managed, so it's always part of business logic. This is very different from the automatic rollback provided by databases for ACID transactions.

Compensation avoids another problem. Locking of your company's data by anyone on the Internet allows denial-of-service attacks. Using compensation means that your data isn't locked for a long time, but we can no longer have ACID transactions – at least the Isolation guarantees must be relaxed – because the data is visible between the initial change and the compensation.

In effect, one trades softening of the ACID guarantees for flexibility, safety, and control over one's own data.

Do We Need Web Services Transactions?

Businesses have been doing without Web services (and similar long-running distributed transactions) for a long time. Do we really need them?

In many cases the answer is "no" – for many business needs, a reliable transport (which itself may use local transactions at the ends) plus a simple application protocol will suffice. Many interactions are between just two parties, so general termination protocols are more complicated than necessary. For example, RosettaNet and ebXML Collaborations are inherently two party.

For more complex scenarios, such as interactions managed in a business process environment (see Yaron Goland's article, "The Race to Create Standards," Vol. 2, issue 6), the answer is a qualified "yes" – in part because these are new assemblies of piece parts, and in part because the complexity of systems goes up. I'll return to this question after we look at transaction specifications.

The Specifications – Overview

I've already mentioned ACID transactions and JTA/JTS. The Open Group (formerly X/Open)

Web Services .NET J2EE XML JOURNAL

LEARN WEB SERVICES. GET A NEW JOB !

The Best .NET Coverage Guaranteed!

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 888 303-5252

*Offer subject to change without notice

SYS-CON MEDIA

SYS-CON Media, the world's leading *i*-technology publisher of developer magazines and journals brings you the most comprehensive coverage of Web services

Distributed Transaction Processing XA specification (1991) defines ACID transactions. JTA/JTS were first specified in the late 1990s.

In XML and Web services there are two key specifications: *Business Transaction Processing* (BTP) from the OASIS standards organization, published in May 2002; and *WS-Transaction* with its accompanying specification, *WS-Coordination*, published by BEA, IBM, and Microsoft, in August 2002.

These specifications have two key concepts in common: compensation instead of rollback, and the use of business logic to determine success or failure rather than all-or-none.

OASIS BTP

OASIS BTP provides two types of transactions. A*toms* resemble ACID transactions in that the result is all-or-none – all of the participants succeed, or the transaction fails and the participants compensate for any completed actions.

C*ohesions* are more flexible. A central coordinator reviews the status of each member of the transaction. Even if some of the members cannot successfully commit the transaction the coordinator can still decide to allow the remaining members to commit. In an ACID transaction such partial success is failure, and all participants must roll back.

For example, you might tentatively reserve rooms at several hotels while building an itinerary, but at the end you only need one hotel for each night, and one flight to the destination. So even if attempting to reserve a room fails at all but one hotel, the transaction can still succeed (see Figure 1).

WS-Coordination and WS-Transaction

This family of two specifications (three, if you count the subparts of WS-Transaction separately) covers much the same territory as BTP cohesions, but adds ACID transactions with Web services as the transport.

WS-Coordination factors out the management and propagation of transaction contexts from the WS-Transaction family of protocols. Context management is a key part of transaction systems. Factoring it out makes it easier to create additional protocols on top of WS-Coordination.

WS-Transaction has two largely unrelated subparts, atomic transactions (AT) and business activities (BA).

WS-Transaction/AT defines ACID transactions using Web services. The goal is to allow interoperability with older ACID-based systems within an enterprise – not across enterprises, for all the reasons we've discussed.

WS-Transaction business activities are another thing altogether – they resemble BTP cohesions, but are tailored as an implementation infrastructure for Business Process Execution Language for Web Services (BPEL4WS), also published in August 2002. Work on BPEL4WS is continuing in an OASIS Technical Committee (see References).

BAs are similar to BTP cohesions: compensating actions are used to undo partially completed work. Business logic (as with cohesions) or a defined business process determine the success or failure of a particular BA.

BAs share the terminology and model with BPEL; the complexities of BA mirror the complexity for an execution language for business processes. In fact, the "BPEL use case" of cross-enterprise business processes and collaboration, as and when they achieve critical mass, are the most compelling use case for WS-Transaction.

Time will tell how rapidly and broadly the adoption of BPEL proceeds, which in turn will drive the adoption of supporting standards.

Future Standards

In the short term, WS-Coordination and WS-Transaction should be submitted to a standards body to start the progress from public specification to standard.

For the longer term, value will be driven by the possibility of legacy integration across operating systems (in the case of AT), and will likely be driven by the attraction of business process integration, management, and execution (in the case of business activities, BTP, or their successors).

At press time, a set of specifications (Web Services Composite Application Framework) was published and promised to be submitted to a standards organization. These overlap with BTP, WS-Transaction, and WS-Coordination. This publication may further accelerate convergence and standardization.

Conclusion

Transactions between enterprises, and support for complex business process execution have driven consideration of technologies that relax some of the ACID transaction guarantees that programmers have used for many years. The need for compensation, rather than rollback, is one basic difference.

The other change is in definitions of success. All-or-nothing semantics work very well in most cases, but the notion of success for a transaction has begun evolving to mean success of a business process, rather than all parts of a transaction are complete.

The WS-Transaction and BTP protocols are among the first formalizations of this new approach, and have similar models. The next months should see the start of standardization of the newer WS-Transaction and WS-Coordination protocols. The widespread adoption of such protocols will depend on the rapidity of adoption of business process environments.

BEA cofounded OASIS BTP, and is a coauthor of the WS-Coordination and WS-Transaction public specifications. We look forward to continuing to drive convergence and broadly accepted standardization in these emerging areas.

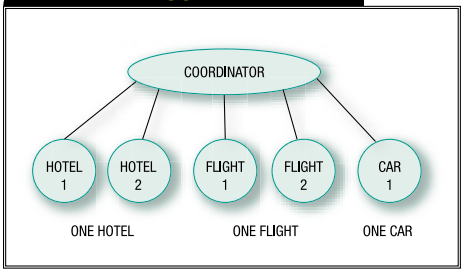
– continued on page 46

TABLE 1

Atomic	**The transaction will succeed completely or fail completely**
Consistent	**The data store changes over time through a sequence of consistent states**
Isolated	**At any instant, an outsider can see whether each transaction has completed (and its results are visible) or not**
Durable	**The results are guaranteed to be stored after the transaction completes**

ACID transactions

FIGURE 1



Transaction Tree, Coordinator

THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany collaboration on a global scale

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

SPECIAL OFFER! SAVE $31*

OFFER EXPIRES NOV 30, 2002

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

Go Online & Subscribe **Today!**

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

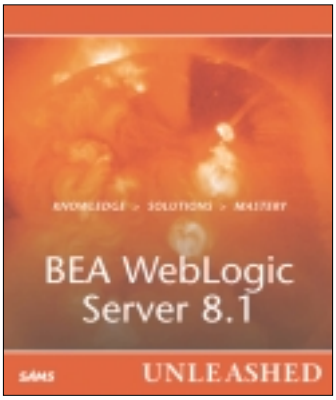
SYS-CON MEDIA

WebLogicDevelopersJournal.com

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

The Role of Programming, Presentation, and Database Skills in WebLogic Server Development

Excerpt by Steve Steffen



EXCERPTED FROM *BEA WEBLOGIC SERVER 8.1 Unleashed*, chapter 3, by Jeff Marin, Steve Steffen, et al.

PUBLICATION DATE:
October 2003

ISBN: 0-672-32487-3
Sams Publishing

The skills needed to build an enterprise application with WLS 8.1 vary somewhat depending on what's being built. However, certain key skills should be present on every development team. Proper thought given to the programming, presentation, and database development skillsets can mean the difference between successful and problematic application deployment.

Java Skills

Java skills are, of course, the essence of WLS 8.1 development. Java skills are vastly different depending on what you're doing. A programmer used to doing small command-line programs in Java won't necessarily even know where to start developing an online application using WLS.

Because of this, assembling a team to build applications becomes more difficult. A manager cannot simply hire a Java programmer; she must know what underlying technologies the application will be using.

The norm in software development is to have one person wear many hats and do multiple things. A word of caution: A developer doing too many things at one time can directly relate to nothing getting done on time. A division of labor and a hierarchy of Java skills are what it takes to develop mission-critical software quickly, efficiently, and correctly.

A Java Architect

Software solutions are only as good or as bad as their design. A Java architect developing for WLS 8.1 must understand all the technologies available and choose the best ones for the solution. Most architects will use a design product to lay out the project – first at a high level, and then at a more detailed level, showing what technologies and classes are being used, extended, and created.

Changes will ultimately come into play as new requirements and technical limitations are uncovered. A good Java architect will have years of experience in software design and an excellent understanding of not only OOP, but the J2EE architecture as well. The architect should also be aware of the features of WLS 8.1, which will enable him to take full advantage of its benefits. This enables managers and architects to work together to provide the leadership and direction for the software project.

JSPs and Servlets

Almost all applications running on WLS 8.1 will use either JavaServer Pages (JSP) or servlets. JSPs include HTML with tags that denote Java coding, whereas servlets are a bit more involved because they're coded completely in Java.

JSPs are the easiest J2EE programs to write. A basic understanding of the Java programming language and certain J2EE-specific Web classes are all that's needed.

Another important part of writing JSPs is writing custom tag libraries to go along with them. Tag libraries enhance the ability of the JSP creator to create complex pages without writing any Java within the JSP file. Tag libraries are written completely in Java and have basic rules associated with them. Java developers should understand how tag libraries work and how to write them.

Servlets are slightly more difficult, but not much. Different levels of Java developers should exist here, with the more advanced programmers leading the entry level programmers.

JDBC and SQL

A J2EE application will usually connect to a database of some kind. This requires the use and set up of Java Database Connectivity (JDBC). JDBC enables Java programmers to connect, create, update, and retrieve information from databases that have JDBC drivers. Structured Query Language (SQL) statements are formed, and data is returned.

JDBC calls are an integral part of the J2EE framework. A developer who works with JDBC must understand SQL and how it operates because improperly designed SQL statements can have a huge effect on the runtime of an application – especially when the. A developer experienced with

connecting with databases is desired here, or a person who understands SQL is needed to properly formulate the SQL statements for the Java programmer.

Note: Developers can always get help with database administrators or designers when constructing SQL statements for a program. However, understanding SQL at a high level is always beneficial.

Beyond the Basics

The scalability of the J2EE architecture is a paramount reason why J2EE is the platform of choice to develop on. Certain technologies work together to give the application seamless integration, even though it might be across different networks on multitudes of separate platforms. The skill set needed for this type of application requires the knowledge of some of, if not all, the following technologies:

- Enterprise JavaBeans
- Container-Managed Persistence
- Bean-Managed Persistence
- Java Message Service
- Java Transaction API and Java Transaction Service
- Remote Method Invocation
- Java Naming and Directory Interface
- Java Management Extensions
- XML Document Object Model (DOM) and Simple API for XML (SAX) parsers
- XML Web Services
- JCOM – Java connector for connecting to Microsoft COM components

These core technologies are advanced and are usually performed or directed by a senior Java developer. One person is usually not proficient in all the skills needed, but two or three developers with overlapping skills can work together to create and deploy these services.

Applets

Java applets are used to fill in the gaps between what HTML can do and what regular applications can do. Applets are graphical in nature, so they comprise a different skill set from J2EE. A J2EE programmer won't necessarily know even where to start when building an applet. Applets aren't used in all J2EE applications, but if dynamically displayed information is needed without requesting it from the server, this is a helpful skill.

Visual Presentation Skills

A picture might be worth more than 1000 words when putting together a Web site. Even if the Web site works properly, graphical enhancements and layouts attract new users and give a professional appearance to the site. Depending on what's being presented, many different skills are needed. The minimal skill set is HTML.

HTML Development

HTML development doesn't require the same technical expertise as coding in Java, but it isn't necessarily easy. J2EE applications have both dynamic and static pages. The dynamic pages are written in JSP and servlet format, whereas the static pages are plain HTML.

Dynamic HTML (DHTML) is an extension of the HTML scope. Using JavaScript, HTML objects can be manipulated to form menus, for example. Also, cascading style sheets (CSS) files can be used in conjunction with HTML to provide a changeable look and feel. A good HTML developer has experience in laying out forms, working with specific tag properties, working with DHTML, and knowledge of CSS tags. There's a difference in how different browsers and versions of browsers display information. A good HTML developer knows these differences and implements browser-neutral applications.

It's also necessary to point out that senior Java developers could have adequate skills for HTML and design, which is helpful. But in my experience, they don't necessarily create the most attractive sites using and watching out for any nuances the way a dedicated HTML person would.

JavaScript

JavaScript is useful in applications by providing a method to access form data and manipulate HTML objects. JavaScript was designed as an easy-to-use component for HTML developers. Consequently, most good HTML developers know this information.

GUI Design

Graphical user interface design is the key component to application usability. Most of the time, the designs change several times before a product is released. The actual programmers sometimes work with the users to find the screen layouts and functionality. This isn't always the best idea. Users are normally used to the old way they did things, so instead of improving the process or using a new layout that might be easier, they want exactly what they had before.

A better choice is to use a GUI designer who understands these concepts and com-

ponents. Sometimes it's not feasible to hire one because of budget constraints. In that case, the developers must be trained in GUI development. Good GUI design can make or break an application because the GUI is what the customer sees first. The customer won't see all the work that was put into the back end of an application, and applaud you for that design, until he can easily learn, find, and use the application's front end.

Graphic Arts

The days of presenting green screens full of data are gone. The presentation of data is now expected to please the senses. In a WLS application, not only must data be created, but pictures, movies, sounds, and other multimedia must also be created and distributed. Engineers don't usually have the skills to create and manipulate this art (they're too busy reading technical manuals). A graphic artist is now becoming a must in the development of applications. The graphic artist works with the GUI designer to create visually appealing screens. The skill of the graphic artist obviously depends on the applications you're building. The more visually intensive your applications are, the better the artist must be.

Database Skills

Database skills are essential for almost any application using WebLogic. Information must be gathered, stored, and recalled on a moment's notice free from errors. Database specialists ensure that the system is scalable and has good performance for the application. This first step in this process is always database design.

Database Design

Database design encompasses knowledge of grouping related information. This enables information to be found and queried easily. Using certain rules such as third normal form, tables, fields, primary keys, foreign keys, and so on can be identified. A good database designer can make it easy for administration later on.

Note: A database designer can make the job of the entire team easy or near impossible. This can affect the entire project development process and even maintenance later on.

Database Development

Database developers take the design specifications and create the database, stored procedures, and views necessary for the application. Each database is slightly different in the features it supports, so database developers should be chosen who usually develop for only one database, such as Oracle or Microsoft SQL Server. This might not be true in all cases, but it's generally true that the larger the project, the more specialized the people.

Database Administration

Database administration is often overlooked when developing applications, but it's a crucial aspect of all areas of an application life cycle. Database administrators are responsible for analyzing database use to increase performance using database views, memory allocation, table size allocation, and stored procedures. These performance tweaks allow WLS applications to speed up the most time-consuming operation: waiting for data.

Summary

Programming, presentation, and database skills are important to the design and implementation of a WLS 8.1 application. These skills depend on the scope of project, and can be covered by a few individuals with a wide array of knowledge or many individuals with more specialized knowledge bases.

– continued from page 42

References

- Andrade, Juan, et al. (1996). *The Tuxedo System: Software for Constructing and Managing Distributed Business Applications*. (Addison-Wesley)
- Gray, Jim; and Reuter, Andreas (1993). *Transaction Processing: Concepts and Techniques* (Morgan-Kaufmann)
- The Open Group (1992). *Distributed TP: The XA Specification*. Free PDF: http://www.opengroup.org/products/publications/catalog/c193.htm
- *OASIS BTP Technical Committee:* www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction. For the specification and primer follow the Documents link at the upper right.
- *WS-Coordination 1.0 (August 2002):* http://dev2dev.bea.com/technologies/webservices/standards.jsp
- *WS-Transaction 1.0 (August 2002):* http://dev2dev.bea.com/technologies/webservices/standards.jsp
- *OASIS Web Services Business Process Execution Language (WS-BPEL) Technical Committee:* www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- Cox, William; and Dalal, Sanjay. "Web Services and Transactions," OMG Workshop on Web Services and the Integrated Enterprise (Philadelphia, April 2003). Earlier version online at www.omg.org/news/meetings/workshops/WebServEurope_Manual/04-2_Cox.pdf

Getting a Handle on Rogue Transactions and Execute Threads

PRESERVING SPACE AND SPEED

BY **LEWIS CIRNE**

Q. What can I do about "rogue" transactions that are affecting the overall performance of my application?

A. Many production J2EE applications suffer from rogue transactions. A rogue transaction is a particular use case or click-through in the application that results in enormous resource consumption or unusually high response times when compared with its peers. In practice, an application suffering from such a problem will display erratic and unpredictable resource consumption patterns or response times.

If we take an application with some single point of dispatch (e.g., controller servlet), it might display two very distinct response times and throughput patterns if there are one or many threads involved in a rogue transaction. Before the rogue transaction hits the server, the CPU and memory usage patterns may be calm. Like a large rock hitting the surface of a pond, from the moment a rogue transaction has entered the application and begun to be dispatched, its voracious CPU and memory consumption may ripple through the rest of the JVM and hardware, perturbing the other, better-behaved transactions in mid-dispatch.

If the rogue transactions are rare, they can show up in performance metrics as sustained anomalies in response time and throughput. If there are a few consistently running at any given time, however, response time and throughput may consistently show poor numbers. In this case, you may not have rogue transaction so much as a couple of consistently expensive use cases. In the latter case, you can generally identify the problem with a profiler or any other tool that will be able to breakdown the responsiveness of the application by application logic or use case.

But what of the situation when you truly have a rare rogue transaction that traverses your system, wreaking havoc in its wake and disappearing as suddenly as it appeared? Your only real option before resorting to tools is to thread dump the JVM while the rogue transaction is being dispatched. Much has been written about collecting and reading thread dumps. You should try to perform at least three thread dumps while your server is in its exhausted state, reading them for any thread that appears to be busy in the same program logic across all three.

Now you should focus your response time analysis on this piece of program logic. If you are sure you can reproduce the same datasets being used by this logic in a staging area, try profiling the logic and measuring its CPU and clock time outside production. If not, consider adding logging or other tooling to monitor the responsiveness of the routines that appear to be costly and slow. One common source of rogue transactions that burn CPU is XML parsing and transformation – some implementations will generate enormous numbers of temporary objects in the course of doing their work, which in turn forces more object instantiation and garbage collection. A cousin of the rogue transaction is the infinite loop: the use case or click-through causes a thread to enter a loop without a break condition. Infinite loops tend to be easier to diagnose. An application server or JVM with a thread in an infinite loop will cause one CPU on the server per thread stuck in a loop, to sit saturated at 100% utilization. The loops should jump out in a series of thread dumps – a thread will literally not have moved from a particular piece of application code in any of them.

Q. What are execute threads and how do I configure them appropriately?

A. If BEA WebLogic Server were said to have any single "utility" it required in order to handle incoming transactions, you might pick execute threads.

One of the main ideas behind J2EE for Web application development was the notion of a servlet container living in a live JVM. With the JVM as a process always resident in memory, incoming requests could be handled by invoking lightweight threads. Even better, you could pool these threads and reuse them when you were finished dispatching a request.

Naturally, like most pooled resources, thread pools have some settings for their initial size and maximum size. They may also have settings for how quickly to grow the pool, how often to check that resources have been returned to it, and so on.

AUTHOR BIO

Lewis Cirne, founder and CTO of Wily Technology, invented the company's core patented Java Agent technology. As Wily's CTO, Lewis takes a leading role in the future technological development of the company and its products, with the goal of extending the services offered by Wily to customers deploying high-performance e-business solutions.

CONTACT...

asklew@sys-con.com

As always, I invite you to send an e-mail to asklew@sys-con.com if you have any performance-related questions about JVMs, Java applications, WebLogic Server, or connections to back-end systems.

In the BEA WebLogic Server, the Execute Queue functions as the thread pool for incoming requests. If you see your response times slow down dramatically as you increase load on your production application – even though you haven't maxed out the CPU(s) of the hardware on which your application is hosted – then you may be bottlenecking your WebLogic Server with its Execute Queue.

A good setting strikes a suitable balance between the available CPU resources and the volume of incoming requests. If your CPU is already saturated, increasing the thread pool size is not going to help the situation and may even make it worse. Larger thread pools mean more work for the thread scheduler in the JVM and for the WebLogic Server itself. The best way to set the Execute Queue's size is to keep an eye on the following four variables:

- CPU utilization
- Application responsiveness
- Application throughput
- WebLogic Server Execute Queue threads busy

BEA WebLogic Server publishes a metric, via JMX, showing how many of the execute threads are busy. If you have the luxury of a staging environment with a load similar to your production environment, try setting the Execute Queue size all the way down to five available application threads – then monitor the business of these threads from the WebLogic Server JMX metrics or some other tool. Slowly increase the number of threads in increments of five, watching how this affects the application responsiveness, throughput, and CPU utilization.

Once you have a thread pool that is staying more than 80% available without the CPU being saturated, you should stop increasing the pool size. Any further increases and you can end up slowing everything down.

News & Developments

BEA Customers, Partners and Developers Embrace Solution to Business Integration Challenge

(San Francisco) – BEA Systems, Inc., the world's leading application infrastructure software company, has delivered on its innovative vision by expanding its product line with BEA WebLogic Platform 8.1, the next-generation solution for enterprise computing that both converges and simplifies application integration and development.

With the industry's first converged application platform suite, BEA delivers superior business integration through a simplified, standards-based application infrastructure that can provide customers with business flexibility, helping drive immediate and future cost savings. www.bea.com

Blue Titan Eases Enterprise SOA Adoption

(San Francisco) – Blue Titan Software, Inc., has announced that its Network Director Web services networking product is now fully integrated with BEA WebLogic Platform 8.1.

Blue Titan Network Director for BEA WebLogic is the only Web services networking product built exclusively for the WebLogic Platform. The joint solution is designed to enable companies to pragmatically adopt enterprise service-oriented architectures (SOAs). www.bluetitan.com

Actional Leverages BEA WebLogic Platform 8.1

(Mountain View, CA) – Actional Corporation has announced that it will leverage the power and performance of BEA WebLogic Enterprise Platform to provide BEA customers with robust Web services management solutions. Actional's Web services management platform enables companies to actively monitor and control highly complex and dynamic networks of Web services, keeping these service networks up and running. www.actional.com

Flamenco Networks to Support WebLogic Platform 8.1

(Atlanta) – Flamenco Networks, a provider of Web Services Management (WSM) software, has announced that their flagship product, Flamenco WSM, supports BEA WebLogic Platform 8.1. Flamenco WSM is designed to make it possible to noninvasively secure, monitor, provision, and manage Web services–based transactions behind and across firewalls. www.flamenconetworks.com

BEA and Siebel Systems Deliver Solutions for UAN

(San Jose and San Mateo, CA) – BEA Systems and Siebel Systems, Inc., a provider of multichannel e-business applications software, have announced availability of Siebel Business Integration Applications for the communication, media and energy industries on BEA WebLogic Integration 8.1. Siebel will also support J2EE standards on WebLogic Platform as part of its next-generation architecture.

Siebel Business Integration Applications are packaged software that integrates business processes across disparate applications and are part of its Universal Application Network (UAN). www.siebel.com

BEA WebLogic Platform 8.1 Chosen by Manugistics

(San Jose, CA and Rockville, MD) – Manugistics, a provider of supply-chain management solutions, has chosen BEA WebLogic Platform 8.1 as its preferred application platform suite. Manugistics' decision was based on a comprehensive review of many application infrastructure products on the market. By selecting BEA, Manugistics can provide customers with the flexibility to support their unique business processes, while reducing costs, enhancing margins, and increasing revenues. www.manu.com

ReportingEngines Offers Application Data Reporting Tool

(Overland Park, KS) – ReportingEngines, a division of Actuate Corporation and a provider of embedded reporting solutions for the J2EE platform, has announced that the Formula One e.Report Engine now includes the ability to access and generate reports from in-memory Java objects. This functionality, known as Application Data Reporting, improves data access efficiencies and increases report system scalability. It also supports a distributed application architecture and provides developers with another data source for reporting. www.reportingengines.com

TrueLink Implements BEA WebLogic Platform 8.1 SOA

(San Francisco) – TrueLink, a provider of consumer credit management solutions and majority-owned by TransUnion, has selected BEA WebLogic Platform 8.1 as its foundation for a next-generation infrastructure that utilizes a service-oriented architecture to accelerate the introduction of new credit-rating products and services that are channeled through leading banking and financial institutions.

Essential to TrueLink's decision to build its entire infrastructure on BEA is the unified development environment provided by BEA WebLogic Workshop 8.1 and the ability to build its applications on one simplified platform that can connect and extend with banking and financial partners. www.truelink.com

Panacya Announces Support for BEA WebLogic Platform 8.1

(Columbia, MD) – Panacya Inc., provider of e-business management software for providing real-time availability and performance analysis across a company's entire Web application infrastructure, has announced support for BEA WebLogic Enterprise Platform 8.1.

Panacya's BusinessAware software suite provides customized support for enterprises deploying the WebLogic Server as well as other common application infrastructure components found in J2EE environments. www.panacya.com

Wily Technology

www.wilytech.com

Quest Software

http://java.quest.com/jcsv/wldj