WEB SERVICES & ENTERPRISE APPLICATION ARCHITECTURE

BEA WebLogic DEVELOPER'S JOURNAL

DECEMBER 2002 - Volume:1 Issue:12

Sitraka

www.sitraka.com/jclass/wldj

BEA Systems

http://dev2dev.bea.com/useworkshop

BEA WebLogic DEVELOPER'S JOURNAL

EditorIAL ADVISORY BOARD
TYLER JEWELL, FLOYD MARINESCU, SEAN RHODY

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JASON WESTRA

EDITORIAL DIRECTOR
JEREMY GEELAN

EXECUTIVE EDITOR
GAIL SCHULTZ

EDITOR
NANCY VALENTINE

ASSOCIATE EDITORS
JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR
JENNIFER STILLEY

WRITERS IN THIS ISSUE
ALI AKBAR, KEN BRADY, PACO GOMEZ, PETER HOLDITCH, NICK MAIORANO, DWIGHT MAMANTEO, PAUL PATRICK, DIANA REID, VADIM ROSENBERG, KEYUR SHAH, FRANK SINTON, JASON WESTRA

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 A. KIRCAALI

COO/CFO
MARK HARABEDIAN

VP, BUSINESS DEVELOPMENT
GRISHA DAVIDA

SENIOR VP, SALES & MARKETING
CARMEN GONZALEZ

PRODUCTION CONSULTANT
JIM MORGAN

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

ADVERTISING ACCOUNT MANAGER
MEGAN RING-MUSSA

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

VP, SYS-CON EVENTS
GRISHA DAVIDA

CONFERENCE MANAGER
MICHAEL LYNCH

REGIONAL SALES MANAGERS
MICHAEL PESICK • RICHARD ANDERSON

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

CUSTOMER SERVICE REPRESENTATIVE
MARGIE DOWNS

JDJ STORE MANAGER
RACHEL MCGOURAN

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

SYS-CON MEDIA

Monsters of the J2EE Gridiron

BY JASON WESTRA
EDITOR-IN-CHIEF

My friends arrived in town (Denver, CO – U.S.) last weekend and to their surprise, I told them I had four football (American football, that is…) tickets to the Broncos game on Sunday. That morning, we proceeded to tailgate, drink, and eat merrily; and then we entered the new "Mile High" stadium to watch our team trounce the unwitting opponent. My mind works overtime, as my wife would say, and as I watched the game it occurred to me that I was drawing comparisons to work-related matters rather than admiring the Broncos. In fact, I felt as though I were watching the future of software and hardware technologies, the "grid," in action. The American football field is often called the "gridiron" because numerous lines and zones divide it. The lines are helpful to players, who use them to determine where they should set up for the next play, or where they need to move during a play.

This is similar to how grid computing works. Grids aren't a new concept, but they have recently gained interest in the computing world for their ability to be self-managing and self-healing, and to automatically distribute computations across a cluster of processors. According to Cindee Mock, director of Competitive Intelligence, Sun Microsystems, Inc., "Grids create a virtualized system of networked resources, easing access for users and simplifying workflow and systems management." Sun has stiff competition from IBM, Dell, and HP, who partnered in September with BEA Systems to sell bundled hardware and software.

Essentially, an optimized grid-computing environment allows your applications to utilize every available ounce of CPU in the grid. This has major implications for J2EE applications, which are typically designed with performance, scalability, and fault-tolerance in mind. Imagine being able to lease space on a grid from your hosting provider's J2EE server farm on an "as-needed" basis. Your application would be monitored externally, or perhaps internally, automatically triggering the activation of itself onto another space in the grid.

The time taken for multiple users to perform seat checks at the football stadium would improve dramatically as processing would be spread across the grid automatically. Because ticket order processing for Broncos tickets is high volume during the initial day of the sale, but marginally low from then on, an application selling football tickets would use a grid, or grid-like, concept to scale up fast, then back down, with limited headaches for the application's system administrators.

The grid environment, or engine, allows an application to utilize only exactly what it needs. As grids and J2EE evolve together, numerous things will have to happen in the way of automatic discovery of services, automated deployment of applications (and even application servers), and of course the automated "undeployment" of applications and application servers. Also, to make the use of grids for J2EE applications convenient, managed service providers will host server farms capable of leasing grid time for overwhelmed applications. Hosting providers will bill on numerous models, but most likely on the number of grids used and time spent processing in each grid. Support and setup fees will be nominal since the grid engine will be set up to automatically handle the provisioning of the applications, tracking them, and even billing the customer at the specified time.

It was fun to watch the monsters of the gridiron (a.k.a. Denver Broncos) win, but I am just as excited to see J2EE and grid computing expand jointly in 2003. It may be too early to tell, but I am also betting that BEA and HP will successfully claim victory of J2EE on the gridiron in 2003 as well.

AUTHOR BIO...

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

CONTACT: jason@sys-con.com

ARE YOUR HANDHELDS SAFE IN J2EE HANDS?

NONTRADITIONAL STATE AND SESSION MANAGEMENT FOR NONTRADITIONAL CLIENTS

BY **NICK MAIORANO**

AUTHOR BIO...

Nick Maiorano is a Sun-certified J2EE architect with over 10 years of experience in software development. He is currently a senior developer and co-architect of a wireless, instant messaging application built atop J2EE WebLogic technology.

CONTACT...

nbox@sympatico.ca

As J2EE usage proliferates into new domains, software engineers are realizing that out-of-the-box components cannot be used to solve problems for which they were not designed. One such example is state and session management (SSM) for applications serving wireless devices. We'll look at why traditional techniques don't work and how you can create your own scalable, fault-tolerant SSM solution.

Despite what your stock portfolio says, there is life out there in the high-tech world. One such example is all the software being created for wireless operator consumption. It's becoming more of a wireless world out there and new specifications are emerging. One example is the Wireless Village initiative created by Nokia, Ericsson, and Motorola. Essentially, this joint venture has created a specification for wireless devices. With its emphasis on interoperability, it addresses instant messaging and presence services (IMPS). This feeds into the expanding universe of mobile information appliances and their associated protocols, paradigms, and policies.

Now more than ever, software engineers must think beyond the browser-based client.

J2EE is at a crossroads and facing increasing competition from .NET. Which one of these platforms will dominate the wireless frontier? Is J2EE up to the challenge?

Browser-Based SSM

What's so different about architecting an application serving wireless devices vs. one serving browsers? The big difference is state and session management – or what I like to call "SSM." To fully appreciate this, let's look at how SSM works for good old browsers.

To communicate with servers, browsers use HTTP, which is by nature a connectionless and stateless protocol. With HTTP alone, the server cannot maintain a relationship among the series of requests originating from a browser. There are times, of course, when stateful connections are needed, as in the case of a shopping cart in an online store. Each browser must be identified on subsequent requests in order to access its own shopping cart stored as an object in the server. Browser identity is commonly achieved through the use of HTTP cookies. When a browser visits a Web site, the server's response can include a cookie that contains a unique session ID. The browser will automatically pass this back on subsequent requests. If a browser does not accept cookies for security reasons, the server can use URL rewriting. Essentially, this means that the session ID is encoded right into the URL.

Whichever technique is used, ultimately a stateful connection is established via a session ID, which should have the following properties:

- Be a long alphanumeric string (80–100 characters)
- Be randomly generated (nonsequential)

These properties lead to session IDs that are difficult to guess and make it challenging for a hacker to masquerade as legitimate user. This requires the attacker to hijack the session, by guessing and using someone else's session ID, to gain access to the system.

Once this is in place, we have a mechanism that allows the server to establish browser identification, create a session, and hold state (i.e., the shopping cart) on behalf of its client. These are the fundamentals of SSM. On a small-scale application, SSM can be fairly straightforward. However, on a big-time, highly available, fault-tolerant, and scalable application, this is quite another matter. Let's take a look at some techniques that WebLogic (version 6.0 or higher) uses to fulfill these big-time requirements.

Performance and Scalability

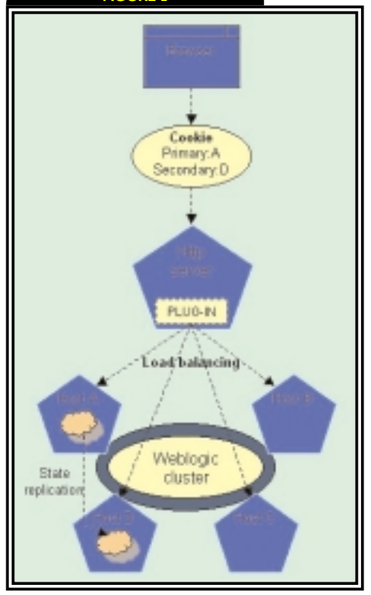
No matter how much CPU power, memory, or disk space is available, a host has a finite load capacity. If the load increases beyond that point, a cluster of hosts is needed to achieve further scalability. This, of course, adds complexity to SSM. Let's look at how this works in a clustered environment. (Note that this is just one of the various configurations supported by WebLogic.)

In Figure 1, the application is hosted by a cluster composed of four identical and interchangeable hosts – meaning a request can be served by any host. Each is a WebLogic server hosting a servlet and EJB container. The cluster is accessed by a dedicated Web server (such as an Apache HTTP server or Microsoft Internet information server) that serves static content. Dynamically generated content, on the other hand, is dispatched to the WebLogic cluster. The Web server contains a WebLogic-specific plug-in (a.k.a. HttpClusterServlet) that load balances each request to a cluster member. Note that a client's session state is stored in only one particular server. Therefore, load-balancing algorithms, such as round-robin, can cause the request to be sent to a host that does not contain the session state. How will session state be accessed in this situation?

Session state could be accessed using network calls (via RMI) but this would incur extra network traffic and reduce performance. Alternatively, replicating the session state on every host so that it is available everywhere would also be nonviable because as the cluster grows, replication becomes more and more costly. N-host replication hinders scalability.

WebLogic has a clever solution. The ses-

FIGURE 1



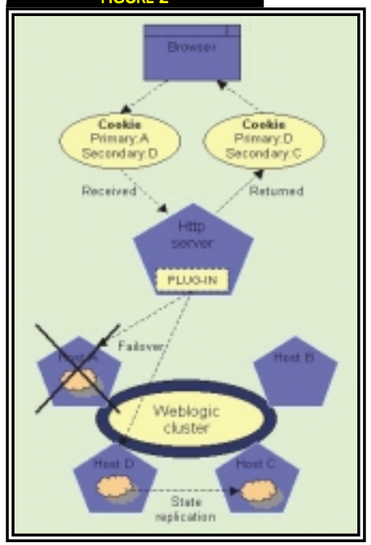
Load-balanced cluster

sion ID, stored in the cookie, has an embedded host name representing the session's home host. The plug-in uses this information to route the request to the home host. This guarantees that accessing the state will be a local, non-RMI call. Typically, processing requests requires many read/writes to session state. When you compare remote calls to local calls, which are 100 times more expensive, this is a key design advantage.

Fault-tolerance and state replication

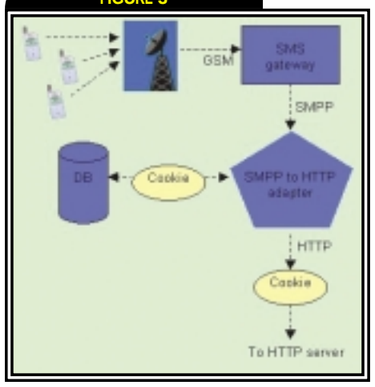
Another benefit of clustered hosts is their inherent fault-tolerant quality. When one host fails, another one steps in to serve the request. All this happens transparently from the client's perspective. Let's look at how SSM fits into this.

FIGURE 2



Fault-tolerant cluster

FIGURE 3



Solution 1: Adapter component

In actuality, a particular session's state is stored on exactly one host (on whichever host the session was created), but is constantly being replicated on exactly one backup host (see Figure 2). Each time an element in the session state is created, updated, or deleted, WebLogic automatically replicates the state on a backup host. Furthermore, to minimize network traffic, only the state delta is replicated.

If the primary (home) host fails, the plug-in will transparently route the request to a secondary (backup) host. Conveniently, all of the session's state will be available and up-to-date right on that host. The secondary will now become the session's primary host and a new secondary will be assigned.

In an effort to avoid n-host state replication, WebLogic employs a one-host replication scheme, or what is referred to as "in-memory replication." As the cluster grows, scalability is not hindered by state replication since there is still only one secondary. The only drawback is that if both primary and secondary fail simultaneously, session data will be lost. WebLogic does offer alternatives, such as database replication, but to the detriment of performance. According to tests, this can represent an 83% drop in performance. In reality, the odds of having two hosts simultaneously fail and losing both session replicas are indeed slim. So in-memory replication is sufficient for most applications.

In retrospect, this is quite powerful functionality. It's all available in J2EE through Sun's servlet specification classes and WebLogic's load-balancing and fail-over mechanisms. Now that I've exposed the inner working of a browser-based client serving, let's see how this relates to wireless clients.

Why Out-of-the-Box Components Won't Do

Unlike the Web browser in the desktop world, there is no ubiquitous counterpart in the wireless world. Wireless devices come in many shapes and sizes and communicate using various protocols. Some are HTTP-based, some are not. All use proprietary networks and all client-to-server communication is mediated by a device-specific gateway. Furthermore, wireless devices must be served differently due to their inherent limitations in screen size, bandwidth, memory, and processing power.

Suffice it to say that despite efforts to make wireless devices behave like desktop browsers (WAP and HDML come to mind), they are a different kind of animal. As far as SSM is concerned, out-of-the-box Web components were specifically designed for HTTP clients. Until the specification addresses this, engineers can choose between two platform-leveraging solutions:

- ***Solution 1:*** Create an adapter component that converts from the native protocol to HTTP – and get all the goodies J2EE has to offer
- ***Solution 2:*** Build upon the platform and roll a custom solution

These can be viewed as "poor man's solution" vs. "rich man's solution" respectively. Don't get me wrong, I'm not passing any judgements in terms of the affluence of these solutions. The first may be adequate for applications that don't require stateful connections. It can also be good enough if the adapter's performance is not an issue. There are caveats, though, and we'll examine them in detail. We'll also look at a best-of-breed solution.

Creating an Adapter Component (Solution 1)

To put things in perspective, let's say you're designing a server-side application that supplies content to SMS-based cell phones upon request. Assume that a stateful connection between client and server is required. The application would most likely use short message peer-to-peer (SMPP) protocol – the most widely used SMS gateway protocol. As Figure 3 demonstrates, the adapter component would convert every request from SMPP to HTTP.

So where do the problems lurk? First, processing requests incurs the extra costs associated with piping requests across two network connections; one between gateway and adapter and another between adapter and application. Second, and more important, there would be two levels of SSM to maintain: the servlet-based SSM in the servlet container and the custom, adapter-based SSM. Essentially, since SMS gateways have no concept of HTTP cookies, this solution requires the adapter to emulate a browser. That is, save cookies and pass them back to the application. However, unlike the browser, which stores cookies for only one user, the adapter would need to store cookies for every connected client.

So, how can we do this? A simple solution is to leverage your database's powerful storage and lookup features to store cookies.

Naturally, extra costs would be incurred with database lookups on every request. Furthermore, the adapter would also be responsible for garbage collection. A custom eviction policy would need to be designed and implemented.

There is another potential side effect. Transactions do not propagate across HTTP boundaries. Therefore, components on either side of the divide could not participate in the same transaction. Consequently, if the adapter would read and lock database tables needed by other components, database deadlocks would ensue. Transactions need to be carefully demarcated.

Alternatively, memory could be used to store cookies. However, with memory, fault-tolerance and replication are now the main concern. If the host fails, the cookie cache must be replicated and available elsewhere.

In summary, this solution can be sufficient for some applications but clearly falls short of servlet-based SSM.

Rolling Your Own (Solution 2)

The custom solution will have the following design ideals:

- Maximize the amount of functionality that can be leveraged from the platform (write as little code as possible)
- Have all the features and performance of servlet-based SSM

Since servlet-based SSM will serve as a model, let's review the major players involved:

- ***HttpSession:*** Holder of session state. Any serializable object can be stored. WebLogic automatically replicates state to a secondary host.
- ***Servlet container:*** Manages the HttpSession cache including the accessibility of HttpSession and garbage collection.
- ***Plug-in:*** Load-balances requests and manages fail over.
- ***Cookie:*** Used by the plug-in to route requests. Contains the primary and secondary hosts for routing.

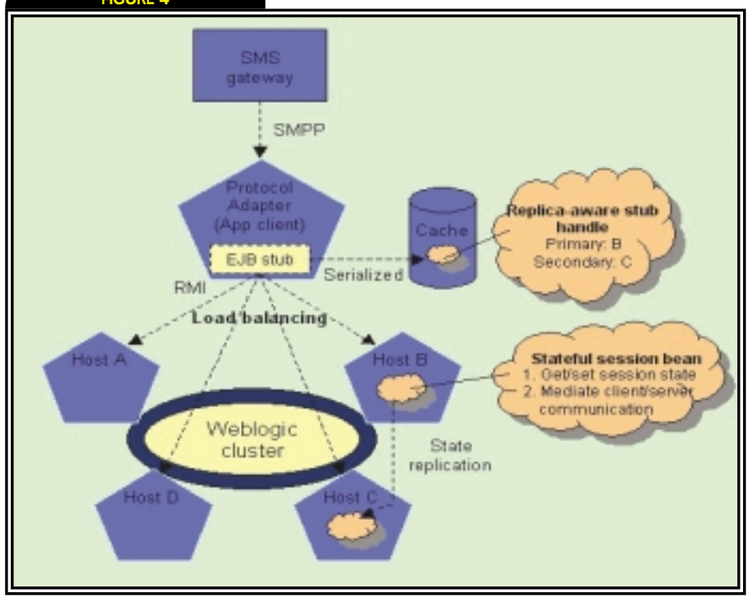
The solution will consist primarily of moving these responsibilities to other J2EE components and writing custom code where this is not possible. In place of the HttpSession will be a stateful session bean (SFSB) in the EJB container. All that's needed is a custom bean class that allows clients to get and set any serializable object representing session state. Note that SFSBs and HttpSessions share the same replication mechanisms. When their hosts fail, and in-memory replication is enabled, their state is available on a secondary host. Unlike HttpSessions, however, SFSBs state replication is controlled by the transactional context. This gives SFSBs a net advantage in keeping session state coherent across the cluster.

In managing the cache of SFSBs, the EJB container is far more flexible than its servlet counterpart. The EJB specification defines the bean life cycle but allows vendors to implement their own cache management schemes. In WebLogic, you can control how aggressively the trash is taken out. First, define a timeout value for the bean. This tells WebLogic when an unused SFSB can be removed either temporarily (via passivation) or permanently (via eviction). Second, choose a cache management policy. Under the not-recently-used (NRU) policy, SFSBs are only passivated when the cache is running out of space. Conversely, the least-recently-used (LRU) policy will passivate upon timeout even if the cache is not full. The timeout value also affects the time of eviction.

As for fail over and load balancing, this will be managed by SFSB's remote stub object. Remote stubs are objects that act as proxies to the actual session bean. Clients invoke methods on these stubs, which in turn invoke on the actual session bean. In WebLogic, stubs are "replica-aware" objects capable of routing the request to the host containing the SFSB. Similar to cookies, they also contain primary and secondary host data used for routing. When a primary host fails, they automatically route to the secondary host and update themselves to reflect the change. If a host fails during a request, however, the remote stub will throw an exception. The client need only retry and the stub will automatically adjust itself and subsequently function correctly.

Like HttpSessions, EJB stubs must be cached and made accessible to clients. Caching can be done by retrieving the "handle" from the EJB stub and storing it in its serialized form. The stub can be re-created at anytime by deserializing the handle. However, since SFSB instances and their corresponding stub objects have different life cycles, the stub cache may not be in sync with the SFSB to which they refer. Making use of the SFSB's call-back methods (ejbActivate, ejbPassivate, ejbCreate, ejbRemove) is the key to keeping the cache in sync with the SFSBs. Figure 4 shows the complete solution using the same application shown earlier.

FIGURE 4



EJB-based SSM

Implementation Caveats

The EJB specification states that a SFSB instance must not be accessed concurrently or an exception is thrown by the container. WebLogic has extended the spec and allows concurrent client invocations via the <allow-concurrent-calls> element in the WebLogic deployment descriptor. When activated, access is serialized at the bean level. This feature only works for SFSBs

–continued on page 40

BEA Hosts dev2dev Days Events Worldwide

MORE THAN 3,000 DEVELOPERS GET AN IN-DEPTH LOOK AT WEBLOGIC PLATFORM 7.0

BY **DIANA REID**

The BEA (http://dev2dev.bea.com) dev2dev Team went global this fall with the first-ever "dev2dev Days" events taking place (with sold-out crowds!) in 11 cities. This one-day, code-level training event offered developers, development managers, and architects a comprehensive look at the new BEA WebLogic Platform 7.0, as well as a peek into some upcoming BEA technology releases. Created by the BEA dev2dev team and BEA Education Services, these events are part of BEA's ongoing efforts to win the hearts and minds of the developer community and ease the complexity of J2EE development. Stops on the tour included Beijing, Boston, Frankfurt, London, Mumbai, New York, Rome, San Francisco, Stockholm, Tokyo, and Vienna, VA. Additional events are being planned in Brazil, Mexico, Israel, The Netherlands, and Washington, DC.

Integration and Web Services

Dev2dev Days presentations and demonstrations were centered around a fictitious company but modeled on a typical, real-world enterprise scenario involving custom application development; integrating legacy systems; connecting with partners, customers, and suppliers; automating manual processes; portal development, and more. Presenters walked through the business drivers – and the cast of characters involved in just about any enterprise IT scenario – for technology infrastructure and change, and then architected, built, and integrated the desired solution from the ground up. The original architecture, laden with disparate and unconnected systems and manual-intensive business processes, was converted to a standards-based distributed system that seamlessly connects a variety of data, applications, and users – and is flexible and easily managed.

The technologies highlighted during the demonstrations included Web services as a key integration enabler; business process management for documenting, modeling, and executing business processes; workflow design, monitoring, and administration; the J2CA architecture for standards-based application connectivity; and portals for "user integration" and personalization. The BEA technologies used included WebLogic Server 7.0, WebLogic Workshop, WebLogic Integration, and WebLogic Portal.

Event presenters were BEA's Tyler Jewell, director of technical evangelism; Michael Smith Jr., principal technologist; and Sam Ramji, Web services architect; each of whom brought their unique enterprise and technologies expertise to the training.

BEA Partners Offer Perspectives

Dev2dev Days featured an impressive lineup of software and hardware vendors who sponsored the global road show. Attendees heard presentations from experts at HP, TogetherSoft, and Intel, and had a chance to see demos and throw hard questions at representatives from these vendors, as well as experts from Borland, Sitraka, and Rational Software over lunch and breaks. The speakers from Intel discussed optimizing performance for WebLogic Server on Intel-based architecture and tuning the WebLogic jrockit VM, and provided a look at the Itanium and Xeon processor roadmap, plus an intriguing preview of new hyper-threading technology. HP's presentation focused on "designing and developing for manageability," emphasizing the importance of building for manageability from the start.

TogetherSoft focused on the tight integration between Together Control Center and BEA WebLogic Workshop, as presenters demonstrated quick and simple steps to build, deploy, and test a Java Web Service leveraging the new TogetherSoft ControlCenter Accelerator for WebLogic Workshop.

SYS-CON Media, Sams Publishing, and *Java Magazin* were media sponsors.

Education, Code, and Free Stuff

Attendees received an impressive array of materials, including WebLogic Platform 7.0 evaluation software, a WebLogic Server 7.0 self-study course from BEA Education and discounts on other training, a copy of ***WebLogic Developers Journal,*** a dev2dev T-shirt, and a CD with all the code samples, demos, and presentations from the event, including a detailed demonstration guide – so developers could test drive the sample application at home. They were also among the first to snag a copy of *WebLogic Workshop Kick Start* from Sams Publishing – the first book to focus on BEA's new Web services development framework and runtime environment.

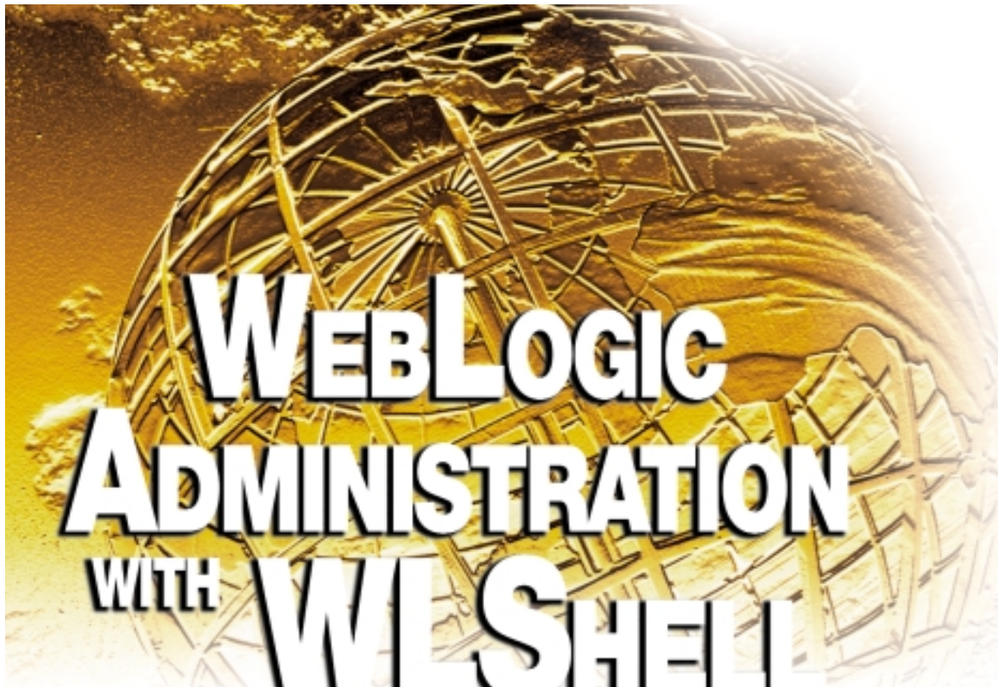
Did You Miss Out?!

To get more information on dev2dev Days, or to review sponsor presentations, visit http://dev2dev.bea.com/dev2devdays.jsp. Developers interested in similar training should contact their regional sales and marketing team for future events or check out the "Introduction to WebLogic Platform 7.0" eLearning course from BEA Education Services (http://education.bea.com).

AUTHOR BIO...

Diana Reid is the senior director of developer marketing for BEA Systems.

CONTACT: dianar@bea.com



COMMAND-LINE SCRIPTING ACCESS TO JMX MBEANS

BY **PACO GÓMEZ**

AUTHOR BIO...

Paco Gómez is principal systems architect at BEA Systems. He designs and develops proof-of-concept architectures for BEA customers and prospects. He is cowriter of the first published book about WebLogic and author of the tool (WLShell) presented here.

CONTACT...

paco@bea.com

Command-line scripting is a well-known and proven approach to managing enterprise software systems. We can find examples of it in operating systems, databases, and LDAP servers. This approach allows the system administrator to run system commands to manage and monitor the system interactively. It also allows you to run commands in batch mode for predefined and repetitive tasks. Is there anything similar in the J2EE world?

While the JMX specification provides an API to manage systems, it doesn't specify how the JMX client application should look. Currently there are two JMX client applications with WebLogic: the Web-based Administration Console and the weblogic.Admin command-line utility. In addition, WebLogic MBeans are well documented, allowing customers and partners to develop their own JMX clients to access WebLogic.

I started developing WLShell to explore this idea of scripting, without programming, WebLogic administration – a tool that would give WebLogic administrators similar features to those well-known administrative shells available in other enterprise systems (like the Unix shell, Oracle sqlplus, etc.). That initial idea is now a tool that is downloadable from BEA's dev2dev site at http://dev2dev.bea.com/resourcelibrary/utilitiestools/adminmgmt.jsp.

What is WLShell? It's a command line interpreter that connects to a WebLogic server and translates user commands to JMX calls. To some extent, WLShell works like the telnet program that connects to a Unix system: it connects to a WebLogic server (administration or managed server) and provides access to MBeans. It supplies the commands "connect" and "disconnect" to manage the connection to a server. Once the connection is established, the user can access MBeans attributes and operations through the WLShell commands "get", "set", and "invoke". The user can then access WebLogic MBeans any number of times under the same connection.

MBean names are usually quite long. To make them easier to remember and shorter to type, WLShell uses a file system analogy. This analogy also fits well in a command-line shell (the Unix shell and Windows command line were designed around the file system). MBean names in WebLogic have at a minimum the domain name, a type, and a name. For instance, a JDBC Connection Pool administration MBean can be named as:

```
petstoreDomain:Type=JDBCConnectionPool,
Name=petstorePool
```

In WLShell, according to the file system metaphor, the domain is the drive unit (as in Windows file systems), the MBean type is the subdirectory under root directory, the name of the MBean is the subdirectory under the MBean type directory, and the MBean attributes and operations are nodes (like files) under the MBean directory. The previous MBean is referenced in WLShell as:

```
petstoreDomain:/JDBCConnectionPool/petstorePool
```

The MaxCapacity attribute of this MBean is referred to as:

```
petstoreDomain:/JDBCConnectionPool/petstorePool/
MaxCapacity
```

The previous notation corresponds to the fully qualified MBean name and MBean attribute name. WLShell also supports partial names by allowing navigation through domains (drives), types, and names (directories) with the "cd" command. The current directory and the directory specified by the command will produce the final name. The following script connects to a server and displays directory contents and the MaxCapacity attribute value:

```
connect localhost:7001 system weblogic
cd JDBCConnectionPool
dir
cd petstorePool
dir
get MaxCapacity
```

The previous general description will give you a basic idea of WLShell. The operation of the shell should appear pretty straightforward and self-descriptive to the user. Table 1 summarizes the commands available in WLShell.

In the next sections, I'll describe three main tasks that can be done with WLShell by accessing WebLogic MBeans: configuring domains, deploying applications, and monitoring runtime attributes. I'll take two scripts included in the distribution of WLShell (in version 1.4) to illustrate those tasks. The first script (petstore.wlsh) connects to a WebLogic server, configures some resources, and deploys the Petstore application. The second script (monitor.wlsh) monitors different runtime attributes of the server, services, and applications.

Configuring Domains

In WebLogic, the general semantic to create services like JDBC Connection Pools is to create the corresponding Administration MBean. This is what the WebLogic Console does when the user configures a domain. As with the other administrative tools, the general procedure to configure a service in a domain is:

- ***Create*** the Administration MBean that identifies the service.
- ***Configure*** the service by setting MBean attributes to specific values.
- ***Deploy*** the MBean to one or more available servers or clusters by invoking an MBean operation.
- ***Save*** the domain configuration to file "config.xml".

The example described here can be executed on a WebLogic domain created with the Domain Configuration Wizard in WebLogic version 7. The script assumes the name of the server is "myserver", but this can be changed in the script. Before starting the server, it is necessary to add the PointBase classes to the classpath. The lines to be added to file startWebLogic.cmd can be found in file pointbase.txt, included with the example files. In addition, the PointBase database has to be started before setting up the domain. The included script, startPB.cmd, will start PointBase. Just review these files to make changes if BEA home is installed on a different directory.

Once a WebLogic Server is running, start WLShell by typing wlsh (or wlsh.sh on Unix) on a different command window. WLShell will first read the commands in the initialization file .wlshrc. This file is used to customize WLShell environment startup. To set up the PetStore application and required WLS services, just type

```
read examples/petstore.wlsh
```

at WLShell prompt. This command will read and execute the commands in the specified file (the script file is under the examples directory). The script does the following tasks:

1. Connects to a WebLogic Server
2. Creates two JDBC Connection Pools
3. Creates three data sources
4. Creates a JMS JDBC data store
5. Creates a JMS server
6. Creates seven JMS Destinations (six queues and one topic)
7. Creates a JMS Connection Factory
8. Uploads and deploys two enterprise applications
9. Saves the domain configuration to file config.xml

After running the petstore.wlsh script, the server should be ready to run PetStore application (http://localhost:7001/petstore/main.screen)

TABLE 1

COMMAND AND ARGUMENTS	DESCRIPTION
$var	Displays the value of variable var.
cd [<directory>]	Displays the name of or changes the current directory.
connect <server:port> [<user> <password>]	Connects to a running WebLogic admin server.
dir [options] [<directory>]	An alias for the ls command.
disconnect	Disconnects from a WebLogic admin server.
domain:	Changes current domain.
domains	Lists available domains.
exit	Exits WLShell.
explore	Opens a WLShell Explorer window.
get [options] (<attr> | <op>)+	Displays the value of the specified attributes or operation information of the current MBean.
help	Provides information about WLShell commands.
invoke <operation> [<parameters>]	Invokes the specified operation with the parameters provided.
mkdir	Creates a directory (MBean)
pwd	Displays the current MBean type or MBean instance.
read <script-file>	Reads and executes the commands in the specified script file.
rmdir	Removes a directory (MBean).
set <attribute> <value>	Sets the specified attribute to the provided value.
set	Lists environment variables.
upload <file> [<app-name>]	Uploads an application file to the server.
var = <value>	Assigns a value to a variable.
ver	Prints WLShell version.
// or #	The rest of the line is a comment.

Available commands in WLShell

Let's take a closer look at task 2, creation of JDBC Connection Pools. The script has the commands shown in Listing 1.

The first four lines will create the Administration MBean that will create the JDBC Connection Pool. Once the new pool is created, we set its properties by setting MBean attributes with the "set" command (lines 5 to 9). In these commands, values of types String and Integer are stored in MBean attributes. One attribute in particular, the "Properties" attribute, expects an object of class "java.util.Properties" that will specify the user to connect to the database (line 7). WLShell provides a mechanism to create an object of a particular class from a String object with a notation similar to Java casting. The command in line 9 will create the expected Properties object.

Finally, the "addTarget" MBean operation is invoked, providing the name of the server Administration MBean where the resource is going to be targeted.

At the end of the first part of the script we can read:

```
invoke $savedom $DOMAIN
```

This command invokes an MBean operation with one parameter to save the domain configuration just created. The fully qualified MBean operation name is in the $savedom variable, which was set up at WLShell start up in the ".wlshrc" file. The parameter is the name of the current active domain, which is stored automatically by the shell in the environment variable $DOMAIN when the user connects to a WebLogic server. The previous command would be equivalent to this one:

```
invoke weblogic:/Repository/Default/saveDomain
petstoreDomain
```

An important point to remember is that in WLShell, all text is case sensitive, including MBean types, names, attributes and operations.

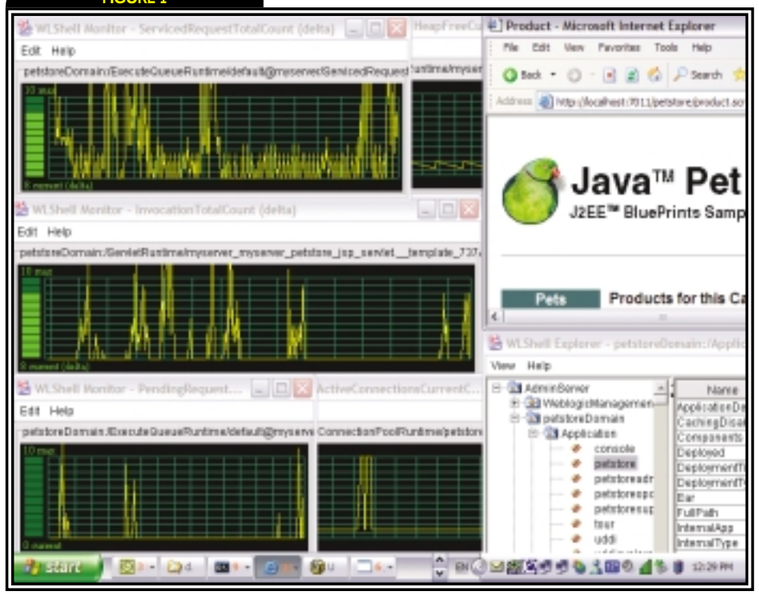
Deploying Applications

WebLogic version 7 introduces a new deployment model that's more robust and easier to use and provides a Deployer component to deploy J2EE applications. This component has an MBean interface that can be found in our example domain at:

```
petstoreDomain:/DeployerRuntime/DeployerRuntime
```

This MBean exposes several operations to activate and remove applications and WLShell provides scripting access to this MBean. In order to deploy an application, WebLogic must have access to the application file or directory (expanded format).

FIGURE 1



WLShell monitoring WebLogic running PetStore

WLShell supplies a command to upload a file to the server file system, very convenient when the administrator has no other access to the remote file system. With these commands we can deploy our application with just four lines (see Listing 2).

The first line creates an object of type DeploymentData with the name of the server where the application will be activated. The second line uploads the local enterprise application file to the server file system. The result of this invocation is the fully qualified name of the file uploaded to the server file system. That result is stored in the $LAST environment variable, as with any other WLShell command. We will need that file name for the activate operation, so we will store it on the $appfile environment variable (line 3). Line 4 is the command that invokes the Deployer MBean operation to activate the application with the required parameters. The application should be ready to use after that.

Runtime Monitoring

WebLogic Server provides another set of MBeans – Runtime MBeans – to monitor runtime attributes of the server, services, and applications. The naming convention for these MBeans consists of the MBean type name ending with "Runtime". As an example, in WLShell Runtime MBeans can be found under the following directories (this is a partial list):

```
/ApplicationRuntime
/ExecuteQueueRuntime
/JDBCConnectionPoolRuntime
/JVMRuntime
/ServerRuntime
```

The second script file in our example, monitor.wlsh, includes a set of commands to monitor some runtime attributes of the server we just set up. To execute those commands, at WLShell prompt type the following command:

```
read examples/monitor.wlsh
```

The script contains the following monitoring commands:

1. A WLShell Explorer, to graphically explore the Domains and MBeans registered in the Administration Server
2. Several WLShell Monitors to graphically display the following runtime values:
   a. Throughput
   b. Queue length
   c. Free heap size
   d. Sockets opened

WebLogic version 7 introduces a new deployment model that's more robust and easier to use and provides a Deployer component to deploy J2EE applications

e. JDBC runtime statistics
f. Servlet runtime statistics
g. EJB runtime statistics
3. Text-based repetitive monitoring to correlate several runtime statistics

Figure 1 shows a screen shot of WLShell monitoring PetStore.

As seen on the script, a WLShell Monitor can be started with the "get –g attribute" command. This will graphically display the numeric value of an attribute over time. Any number of Monitors can be started to display any numeric attribute. This generic approach makes it possible to monitor any numeric MBean attribute in WebLogic, like a custom Execute Queue created in a Domain, runtime attributes of Web, EJB Components, etc.

An additional option to the "get –g attribute" command, the "-d" parameter, will tell the Monitor to display the difference between the current reading and the last one (delta) instead of the attribute value. This convenient flag allows you to graphically display things like throughput of the execute queue based on the number of requests serviced. Other interesting examples of this type of monitoring are the number of sockets opened on a server, the invocation count of a particular servlet, and the number of transactions committed by an EJB.

Server Life Cycle

Server life cycle can also be controlled through WLShell by invoking the corresponding MBean operations. For instance, a server can be shut down with the following command:

```
invoke /ServerRuntime/myserver/shutdown
```

To start a server with WLShell – the operation to invoke is in the corresponding Administration MBean – the command is:

```
invoke /Server/myserver/start
```

While this can be done with just one WLShell command, there are other components that have to be properly configured and running. WLShell invokes the operation of the Administration MBean on the Administration Server, the admin server connects to the node manager running on the machine where the server is located, and finally the node manager starts the server.

Other management operations include pinging the server for availability, suspending and resuming the server, etc. Examples of these operations can be found in WLShell script files under the "scripts" directory.

Moving Forward

A typical domain configuration with a few services and applications, like a WebLogic Portal Domain, can easily have over 1,300 MBeans registered. How can a WebLogic administrator know where to go to manage the domain? There are several suggestions that can be helpful. The first source of information is the WebLogic Administration guide, which provides information about MBeans and management tasks. In addition, the WebLogic Console offers help on every manageable attribute on the console, just click the question mark icon on the left of the attribute name and a help window will pop up providing the name of the MBean that contains the attribute and a short description.

WLShell will also display the description of MBean attributes through the command "get –v attribute" (v stands for verbose) and through the WLShell Explorer ("explore" command).

Summary

WLShell is a command-line shell for WebLogic administration, including domain configuration, monitoring, and server life-cycle control. It provides a generic access mechanism to any MBean in WebLogic, without requiring any Java programming. The simplified, yet complete, notation for MBeans can make a domain configuration and its documentation more concise and less error-prone. It is ideal for repetitive management tasks like setting up services, deploying applications on different servers, and monitoring. It is also convenient for interactive administration.

Listing 1

```
mkdir /JDBCConnectionPool                                          //1
cd JDBCConnectionPool                                              //2
mkdir petstorePool                                                 //3
cd petstorePool                                                    //4
set URL "jdbc:pointbase:server://localhost/demo"                   //5
set DriverName "com.pointbase.jdbc.jdbcUniversalDriver"            //6
set Properties (Properties) "user=petstore"                        //7
set Password petstore                                              //8
set MaxCapacity 10                                                 //9
invoke addTarget /Server/myserver                                  //10
```

Listing 2

```
data = (DeploymentData)"myserver"                                  //1
upload "C:/bea/weblogic700/samples/server/stage/petstore/petstore.ear"  //2
appfile=$LAST                                                      //3
invoke /DeployerRuntime/DeployerRuntime/activate $appfile petstore null $data null//4
```

THE NEW SECURITY ARCHITECTURE OF BEA WEBLOGIC SERVER 7.0

A NEW END-TO-END, POLICY-BASED EXTENSIBLE SECURITY FRAMEWORK MAKES A BIG DIFFERENCE

BY **PAUL PATRICK**
& VADIM ROSENBERG

AUTHOR BIO...

Paul Patrick is the chief security architect for BEA. He was the architect of BEA's (and earlier Digital Equipment Corporation's) Object Broker CORBA ORB, coarchitect of WebLogic Enterprise (Tuxedo), and has been the security architect for WebLogic Server since version 6.0.

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

CONTACT...

vadim@bea.com

This article appears courtesy of BEA dev2dev Online.

Installing and maintaining security is a huge challenge for an IT organization. To serve a worldwide network of Web-based users, the IT organization must address the fundamental issues of maintaining the confidentiality, integrity, and availability of the system and its data. Security across the infrastructure is a complex business that requires vigilance and established and well-communicated security policies and procedures.

This article looks at securing the Java-based application and the WebLogic Server on which it is deployed. WebLogic Server 7.0 incorporates a completely redesigned security architecture that provides a unique and secure foundation for applications. WebLogic Server 7.0 security services can be used standalone to secure WebLogic Server applications, or as part of an enterprise-wide security management system that represents best-in-breed third-party security management solutions.

Security Issues Facing Customers Today

So, what are the problems with security? Well, there are quite a few, but the major ones that we've heard from customers are:

- Application security today is in the hands of application developers. In order to implement really strong security or any kind of business security rules, the security-related code is included in the application. Since developers are typically not security experts, this makes it error prone and extremely costly to develop and maintain.
- Hardcoded security policies are inflexible and policy changes require changes to application code, which is slow and expensive.
- The need to integrate new applications with existing security products usually requires a very costly "custom code" to plug into third party products.

Today, customers have to build aspects of application security directly into their applications. By building proprietary connectors, they can utilize the third-party point security solutions directly, which of course locks them into a single vendor and proprietary technology. And, if any intelligent business security rules need to be implemented – customers build their own security policy systems. This distracts them from implementing their core business functionality and increases time-to-market immensely.

Why Is J2EE Not Good Enough?

J2EE security attempted to provide a simple infrastructure to solve security issues. However, it turned out that in the real world J2EE security standards aren't strong enough or flexible enough, and in general don't have many of the features required by a modern agile enterprise application. These are some of the problems with J2EE security:

1. Requires developers to hard code security into the business logic and configuration files.
2. Administrators cannot change security settings – they need to know too many things to do it.
3. Developers and administrators cannot implement business rules for security policy – there is no concept of business security rules in J2EE.
4. Only controls certain J2EE components (EJBs, servlets, JSPs), not the entire application (what about JCA, JMS, databases, and all those non-J2EE components like Web services?).
5. Not integrated with the leading security ISV solutions that might be an existing corporate standard – many of these products are not even based on J2EE.
6. Has no provisions for Single Sign-On (SSO).

The Solution: A Security Framework

The WebLogic Security Framework, new in WebLogic Server 7.0, provides end-to-end application security, covering J2EE and non-J2EE components of your application hosted on WebLogic Server. With WebLogic security:

1. Security policies are created and managed by Security Administrators.
2. Security policies are flexible, dynamic, powerful rules that can be changed without recoding and redeployment.
3. Integration with existing security solutions is greatly simplified.

Unlike J2EE, the WebLogic Security Framework separates application business logic from the security code. Security services, including security business rules, are provided by the infrastructure and don't have to be coded in the application. It's easy for nondevelopers to administer and doesn't require any programming or XML knowledge. A GUI for security administration is provided out-of-the-box.

A built-in dynamic security rules engine makes it easy to implement dynamic business rules for security policies, and does not require any downtime to update these rules. It allows mapping company business rules to security policies in distributed deployments, providing easy customization of application security to business requirements.

With an open Security Service Provider Interface (SSPI) the framework allows leading security solutions on the market to plug in and provide their security services to WebLogic applications, and also enables adding custom extensions. In addition, WebLogic Server 7.0 provides prebuilt implementations (security service providers) for most of these plug-in points.

Single Sign-On is automatically available to WebLogic Server applications without any additional programming.

WebLogic Server provides a complete range of security coverage for all J2EE and non-J2EE components deployed in WebLogic Server.

Having said all this, it's important to remember that as a certified J2EE 1.3–compliant application server, WebLogic Server supports all the security features required by J2EE, such as JAAS. Also, it supports the WebLogic Server 6.x security model by providing a "compatibility mode" which should make it easy and painless to transition from the older 6.x security model to a new security framework.

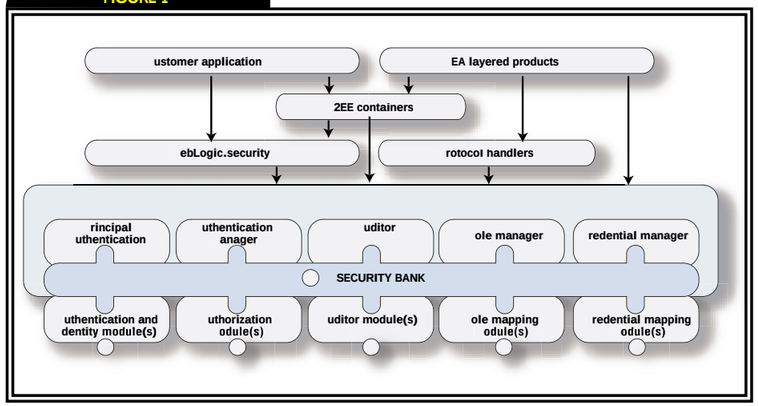
With an open architecture, standards support, and unified administration, WebLogic Server 7.0 security gives the IT department the tools it needs to address real-world issues in security.

Putting It All Together: the New Security Architecture

Figure 1 shows the WebLogic Server 7.0 service-based Security Framework, which provides interfaces to other BEA products, J2EE containers, and customer applications, and delegates requests to the appropriate security plug-in. Security plug-ins supplied by BEA with WebLogic Server perform the following functions out-of-the-box:

- ***Authentication:*** Authenticates, verifies, and maps security tokens to an internal format for security support. Supports delegated username/password and certificate authentication with WebLogic Server, and HTTP certificate authentication via the standard service provided in a Web server.
- ***Authorization:*** Enforces authorization policies for resources, taking business policies into consideration. Supports role-based authorization, in which access is based on job function and business rules.
- ***Auditing:*** Audits all security actions in support of non-repudiation. Provides a customizable set of data for auditing security events such as failed login attempts, authentication requests, rejected digital certificates, and invalid roles.
- ***Public key infrastructure:*** Supports standard public key encryption for data or digital signatures, or when electronic

FIGURE 1



The WebLogic Server Security Framework

authentication of a client's identity is required.

- ***Credential mapping:*** Maps a user's authentication credentials to those required for legacy applications, so that the legacy application gets the necessary credential information.
- ***Role mapping:*** Maps roles to users or groups, based on policy. Determines the appropriate set of roles granted to a WebLogic Server user or group for a WebLogic resource.

The Security SPI: the Interface for Flexibility

The security plug-in scheme in WebLogic Server 7.0 is based on a set of Security Service Provider Interfaces (SPIs) for the plug-in points. The Security SPIs can be used by customers or third-party vendors to develop security plug-ins for the WebLogic Server environment. Security SPIs are available for authentication, authorization, auditing, credential mapping, role mapping, and the public key infrastructure (supporting the Java standard Key Store for encrypted storage of public and private encryption keys).

The Security SPI scheme means that customers have four choices for securing WebLogic Server installations:

- BEA-supplied security plug-ins
- Third-party security plug-ins based on the BEA Security SPI interface
- BEA Security SPIs to create customized security plug-ins for WebLogic Server systems
- Existing third-party security technologies that have been adapted so that they are BEA-compliant (some are available today or are coming in the near future)

An Open Architecture: Multi-Vendor and Multi-Protocol Support

The open, interface-based security architecture in WebLogic Server allows use of existing security products while taking advantage of new security technologies available in the marketplace. With this architecture, a security installation can support security vendors' *full value propositions*, not just a subset. A user's choice of security products can be "mixed and matched" to create complete custom security solutions. In fact, WebLogic Server installation can run *more than one* security plug-in for a given function, and users can set constraints that govern which product or protocol will be used in a given situation.

As users integrate new solutions or modify existing ones, administrators can set *security policy* for each security plug-in, using a built-in menu-driven policy tool. Security policy governs authorization: the rules and constraints for accessing resources or assuming roles. More than one security plug-in can run concurrently, as part of a migration or transition scheme, and set security policy accordingly. The BEA-supplied Adjudicator function resolves any conflicts in interpretation when making authorization decisions.

The WebLogic Server 7.0 design for security services supports any choice of vendors and protocols because it separates the details of the security system from application code, simplifying application maintenance and management. Changing security system components or policies need not entail modifying applications. This unified architecture makes it easy to integrate best-of-breed security solutions, and to replace components of a security system with the latest technologies from third-party vendors, or from a development staff. The ability to swap in new security plug-ins and technologies as needed reduces the total cost of ownership and maximizes the return on investment in security technologies.

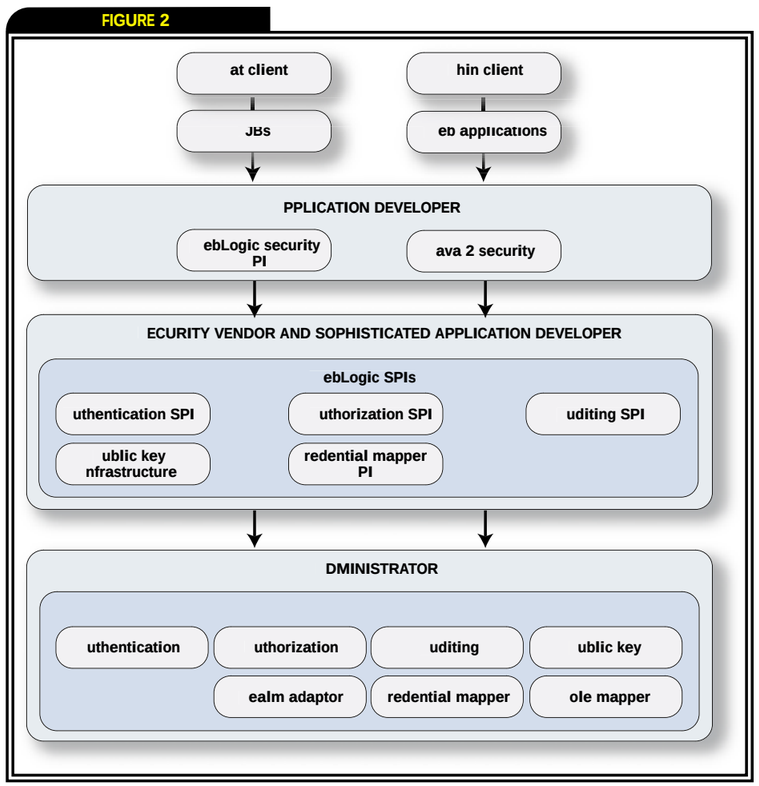
Advantages for Developers, Administrators, and Vendors

Figure 2 illustrates how different users would interact with the software architecture of the WebLogic Server security services. The new security architecture has benefits for three categories of users: application developers administrators, and third-party security service vendors.

Benefits for Application Developers

Since most of the security functionality for Web applications can be implemented by a system administrator, application developers need not pay attention to the details of securing the application unless there are special considerations that must be addressed in the code. In cases where programming custom security into an application is required, WebLogic Server application developers can take advantage of BEA-supplied Application Programming

FIGURE 2



WebLogic security for various types of users

Interfaces (APIs) for obtaining information about subjects and principals (identifying information for users) that are used by WebLogic Server. The APIs are found in the weblogic.security package.

With WebLogic Server's support for the Java standards, developers can also use the APIs in the Java platform security packages such as JAAS and JSSE, as well as the security-specific methods defined by J2EE.

Benefits for Administrators

Administrators who install, configure, deploy, and maintain WebLogic Server can use their choice of BEA-supplied security plug-ins, customized security plug-ins, or third-party security products, and manage them all with the Administration Console.

Out-of-the-box, a complete security solution can be implemented using the BEA-supplied security plug-ins. Administrators can use the menu-driven rule-based policy engine to create an authorization scheme that implements your company's business rules.

SETTING POLICIES: NO PROGRAMMING REQUIRED

The built-in Policy engine provides a GUI interface that lets Administrators set policies in the Administration Console, without writing application code. By right clicking on the system resource displayed in the Administration Console, users can select among the constraints displayed on the drop-down menus. Figure 3 illustrates this simple menu-based approach to adding or changing security in applications.

Benefits for Third-Party Security Vendors

Most leading security service providers have announced plans to support WebLogic Server 7.0. These providers are integrating their products with the WebLogic Server environment using the Security SPIs. As the underlying integration mechanism for BEA's security plug-ins, the Security SPIs permit development of customized security plug-ins for the WebLogic Server environment. Security SPIs are available for authentication, authorization, auditing, public key infrastructure, credential mapping, and role mapping. This allows third-party vendors to provide tightly integrated solutions that are easy to implement.

Security via Users, Roles and Policies

The key to WebLogic Server 7.0's security architecture is the organization of application users into *users* and *groups* that take on *roles* according to defined security policies. Users can be organized into groups. Groups can be used to represent organizational boundaries as well as to simplify administration. Each application user and group is mapped to a role dynamically during application execution, when authorization is needed.

Roles and policies determine access to system resources, and permitted behaviors. User roles are registered by an administrator using the built-in menu-driven security policy tool embedded in the BEA-supplied Authorization plug-in. The security policy tool's interface reflects business concepts, not programming concepts, and allows an administrator to create simple prose-based rules for dynamically assigning roles and calculating access privileges. Application developers are freed from having to write application code to implement complex business policies, because the policy tool separates the tasks of business policy creation and application creation.

The roles that a user can be assigned to are determined by policies defined by the administrator, on behalf of the company. Since policies reflect business security rules, a company's management sets security policies rather than the software development staff. Security policies can easily be changed with changes in business conditions.

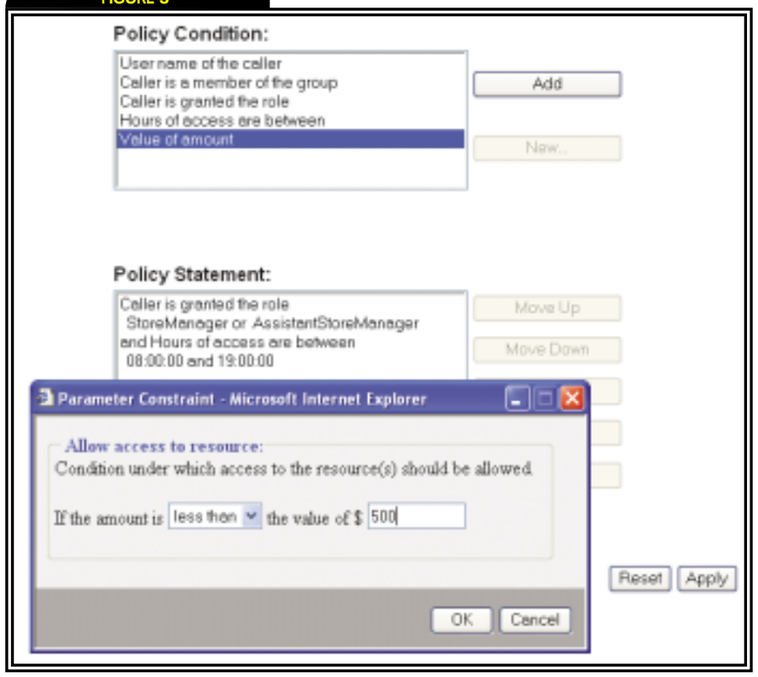
The role-and-policy-based security scheme replaces the previous scheme of users, groups, and access control lists (ACLs), and offers clear advantages for ease of administration and ease of adaptability as security requirements change. Using roles and policies for authorization permits dynamic computation of access status for each resource, for each user or group.

WebLogic Server 7.0's dynamic, role-based authorization scheme can be applied to all WebLogic Server resources. The administrator and applications developer are no longer constrained by the limitations of the declarative security model in J2EE, which embeds security constraints in the code and makes it difficult to modify a security scheme when business requirements change.

• • •

Next month, I'll look at more of the details of the security functionality provided by WebLogic Server 7.0.

FIGURE 3



Authorization Policy sample

The Role of Web Services in Enterprise Application Architecture

MEETING TODAY'S BUSINESS NEEDS WHERE YESTERDAY'S COMPONENT ARCHITECTURES FAILED

BY **FRANK SINTON**

Web services play an important role in building enterprise application architectures. These architectures, by nature, provide a blueprint describing software structure. This blueprint allows enterprise application architecture to be the confluence of business and technology, supporting business requirements and providing technology enablers.

AUTHOR BIO...

Frank Sinton is the president of Digital Coast Solutions (DCS), a full-service technology consulting firm. He specializes in enterprise architecture, Java, and Web services for clients ranging from Fortune 100 companies to startups. Frank has over 10 years of experience in leading technology efforts in enterprise software development.

CONTACT...

frank@digitalcoasts.com

For enterprise application architecture to accomplish this task, Web services will play a critical role. In this article, I will discuss the vision for Web services within complex corporations, how it compares and fits versus EAI, and how to begin implementing Web services within a corporation.

Background

In order to understand the role of Web services in enterprise application architecture, the architecture must be fully understood. Enterprise application architecture is defined as "the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time." With today's business requirement of positive return on investment (ROI), application architects must focus on the business application portfolio to show direct business value. Redundant databases and silo applications produce operational surplus, negatively affecting the bottom line. This is not tolerated in today's business climate. In short, information must be accessed in an easy, comprehensive manner. Thus, enterprise application architecture needs to set enterprise-level objectives for:

- Consistency, integration, interoperability, and security
- Reuse across applications
- Flexibility to change applications

With these objectives, enterprise application architecture can drive high-quality systems – "better, faster, cheaper" – to achieve what every business desires:

- Timely access to data and reports whenever and wherever needed
- Flexible, adaptable systems that respond to rapidly changing business conditions
- Accurate and consistent data throughout every department
- Seamless integration and data sharing across the enterprise
- Dependable security and reliability
- Reasonable and affordable cost

How will enterprise application architecture achieve this? Enter Web services.

Web Services and the Enterprise Application Architecture Evolution

Through Web services, enterprise application architecture has evolved from a component-based architecture to a services-based architecture. Both component and service architectures rely on the same basic concept of reuse. However, a services-based architecture takes a component-based architecture and adds a programming model on top of it. That programming model is Web services.

Web services allow components to be published, discovered, and invoked over the network using standard Internet protocols (primarily HTTP(S) or SMTP). This new services-based architecture hides the underlying technology of a component, allowing an application to use the service without having to understand the underlying technology. An application need only know what a service does, how to use it, and where to find it. Additionally, the loose coupling of Web services allows applications that rely on that service to continue to run regardless of implementation changes. These two key pieces of services-based architecture – technology independence and loose coupling – fulfill today's business needs where yesterday's component architectures failed.

Fulfilling Today's Business Needs

There's certainly been plenty of buzz surrounding Web services technology, with its promises to reduce integration costs, change applications

quickly, and exploit software resources across business units. But how do Web services bring value to business today?

Before you can understand how Web services can be applied in the enterprise, you must understand how large enterprises are organized. Large Fortune 500 enterprises are typically divided into both lines and sublines of business. These "companies within a company within a company" can consist of hundreds of employees with distinct business objectives and supporting applications. A look at today's entertainment conglomerates provides a good example. Sony, Vivendi, News Corporation, and Viacom are broken down into several lines of business, such as electronics, music, entertainment, and online. From here, each line of business is typically broken down into business units – for example, entertainment may be divided into home entertainment, television, and motion pictures. These corporate structures have caused a number of IT problems. The different business initiatives have created silo applications and redundant databases, resulting in poor information velocity and high costs. Web services help solve these problems through:

- ***Application infrastructure services:*** Web services technologies can be used to deliver behind-the-scenes functionality such as user authentication, divisional content sharing, and common product sharing.
- ***Enterprise application integration "Light":*** Web services technologies can solve the complex integration challenges of linking applications across business units within an enterprise, at a cost that can be justified.

Application Infrastructure Services

Using Web services, cross-divisional sharing of application infrastructure services eliminates the need for the "big bang" approach that has plagued corporate-wide initiatives such as single sign-on and enterprise-wide portals. Instead, instant efficiencies can be realized through the development of Web services from existing business components. Developers can provide centralized business components with the simplicity and ubiquity of XML over HTTP.

Web services are ideal for building reusable business components when data and business rules need to be reused between two or more applications. These components can be grouped together to form a business process or stand alone. Once these Web services have been developed, they can be registered in a local UDDI, allowing other divisions to tap into them – no all-encompassing, enterprise-wide effort is needed.

Product rights lookup is a perfect example of an application infrastructure service that can be developed as a reusable business component. Rights are typically addressed on an application-by-application basis rather than an enterprise basis. These rights are managed by different systems and can lead to inconsistent rights information, enough to make lawyers cringe. By developing and deploying a product rights lookup Web service, enterprises can make that information reusable by all applications requiring rights information.

Enterprise Application Integration 'Light'

Enterprise application integration (EAI) platforms are at the forefront of most enterprise application architecture concerns because EAI revolves around understanding information flow and efficiencies. Generally, most large companies looking at EAI solutions fall into three categories:

- ***Companies who have purchased one or more EAI packages:*** But they may still be having trouble making sense of them.
- ***Companies who realize their integration problem:*** Many point-to-point interfaces with very little overall understanding of their integration architecture. These enterprises are exploring ways to transform their integration problem to serve their business needs and produce a positive ROI.
- ***Companies who have a particular project with a complex integration requirement:*** An example of such a project is an Enterprise Resource Planning (ERP) system replacement.

Both Web services and EAI can provide solutions for these three categories of companies. The key question for corporations is which solution is better for what situation?

Web services are ideal for EAI when a rapid, noncomplex integration solution is needed. Web services promote a peer-to-peer type of integration environment that is relatively low-cost to operate and still provides better federation than point-to-point integrations. On the flip side, existing EAI technologies use expensive and proprietary technology to conquer the 20% of projects that are very complex or high in volume. Existing EAI vendors best serve this need. The remaining 80% of integrations are ideal for Web services because Web services are low in cost, standards-based, easy to learn and use, and endorsed by the entire technology industry.

In addition, combining Web services with other technologies such as portals – reusable user interface components – is another area in which enterprises should look to use Web services as an EAI solution. Because diversity is commonplace in corporate environments – it's not unusual to see two or more ERP systems, two or more CRM systems, and two or more portal systems – Web services can make these systems work together. The end result is a shortened development time frame and a more consistent, high-quality product.

Implementing Web Services Within an Enterprise

How should enterprises begin developing and deploying Web services? First, IT organizations need to immediately address two issues before 'Web services creep' gets out-of-hand, much like corporate Web sites in the mid-1990s:

- Who governs Web services?
- What framework controls should be put around the development and deployment of Web services?

Governance of Web services: Enterprise Application Architecture

Because any application developer can build a Web service relatively easily and cheaply, IT governance of Web services is an important issue and cannot be underestimated. Without the right IT governance, a web of applications producing and consuming Web services can exist before an IT executive even has Web services on his or her radar. So who

will own the central registry of Web services, deciding where and when to produce and consume them? In addition, who will monitor these Web services to make sure they perform as advertised? Enter enterprise application architecture.

Enterprise application architecture groups allow the federation of Web services through a centralized group. Since architects tend to also be the visionaries and most experienced technology people, they will be able to make the crucial decisions on when and when not to produce Web services for certain business processes. Furthermore, this will allow for creativity and experimentation with this new technology while keeping it under control. Finally, managing and monitoring centralized repositories of Web services makes sense for enterprise application architecture since the architects' primary responsibility is to unite business and technology to meet business needs.

Web Services Framework: WebLogic Workshop and BEA Platform 7.0

In a Web services world, corporations will continue to head towards the process of application assembly and away from application development. Application assembly involves the typical component architecture practices of reusing frameworks and shared components. With Web services, application assembly additionally involves the services architecture practice of consuming Web services. What tools are available for application assembly of both components and services today? Enter WebLogic Workshop.

WebLogic Workshop provides a visual development environment that lets non-Java developers generate J2EE code simply by visually specifying business processes. Application developers can build applications by adding methods and controls, setting properties, and writing business logic using Java. To reduce tedious Web services work for Java and non-Java developers alike, Workshop also automatically generates WSDL contracts and all related XML code, including SOAP messages, UDDI registry entries, and schemas. Further, WebLogic Workshop validates the WSDL contract, reporting any violation. Loose coupling and both asynchronous and synchronous communications are fully supported.

Java developers can create Workshop controls through Enterprise JavaBeans, direct database connections, or other methods. Additionally, any Web service can be converted automatically into a control on WebLogic Workshop's palette and be made available for reuse. This makes creation of Workshop controls ideal for hard-core Java developers who are focused on key system components rather than new application development.

Finally, WebLogic Workshop provides its own runtime engine through BEA's WebLogic Platform 7.0. This allows for a single, easy-to-use, infrastructure platform to develop, deploy, and manage Web services, and eliminates the need to write the hundreds of lines of J2EE plumbing that it would take to run a Web service without the runtime framework.

Despite its technical achievements, perhaps the most appealing aspect of Workshop for enterprises is its ability to separate the responsibilities of the enterprise developer and the application developer. The enterprise developer, who is part of the governing body of Web services, can concentrate on creating controls and the underlying frameworks, while the application developer can focus on assembling actual business applications. This separation of concerns works well in today's line-of-business organizational structures. Line-of-business developers can focus on their particular business applications while still plugging into the "enterprise plumbing" that is produced by the enterprise developers through Web services.

Conclusion

Web services are the future and if CIOs are unprepared, they're liable to face the same integration and silo application problems tomorrow that they have today, but with an additional amount of complexity through poorly managed Web services. However, CIOs who provide the proper governance through an enterprise-wide group such as enterprise application architecture and establish solid frameworks for Web services such as BEA Workshop and Platform 7.0 should find themselves in a position to drive significant cost savings through applications infrastructure services and application integration.

Reference

- *IEEE:* STD 610.12 (Hagle brief)

Multitiered Performance Evaluation with PerformaSure 1.6

MEASURING PERFORMANCE, EVERY STEP OF THE WAY

sitraka
the Java™ advantage

Reviewed by Ken Brady

Company information
260 King Street East
Toronto, Ontario, Canada
M5A 4L5
1-800-663-4723
E-mail: info@sitraka.com

For sales, service, and presales information:
E-mail:
direct@sitraka.com
Phone: 1-800-663-4723

Test environment:
Windows 2000 environment running BEA WebLogic 6.1

Imagine a commuter rail system that is controlled by an individual standing at the end of the track logging each train as it arrives. Without a comprehensive tracking system that pinpoints each train while en route, it would be impossible for that person to identify and troubleshoot train delays. Instead, that person is left to "shoot from the hip," guessing at the possible cause of the delay. Web applications have the same limitation. Without a tool that offers a comprehensive look at the entire Web transaction, Web developers are left to guess and point fingers, instead of resolving performance issues. That's where Sitraka's PerformaSure 1.6 really shines.

PerformaSure is a performance analysis tool designed for multitiered, J2EE-based Web applications developed in the BEA WebLogic environment. Where traditional performance test tools only identify that the transaction took a long time, PerformaSure identifies which step in the multitiered process is responsible for the slowdown. This helps to eliminate costly misdiagnoses and speeds corrective action. PerformaSure is the ideal performance diagnostic tool for developers and implementers of complex systems with multiple application servers, databases, and Web servers.

PerformaSure is not for everyone. The complexity of installation and configuration and the intense resource requirements will be too cumbersome for smaller shops. Also, PerformaSure requires the use of software agents that are tightly integrated into each application and Web server along a transaction's path. This may make enterprise developers leery about installing this tool on production systems. Still, in development and preproduction environments, PerformaSure will provide invaluable information as developers shake out the performance bugs in their applications. And the more complex the application, the more valuable PerformaSure becomes.

Components

PerformaSure is comprised three main components: agents, the Nexus, and workstations. The PerformaSure Agents instrument the operating system and applications on each server. The agent is a vehicle whereby PerformaSure monitors various performance metrics of the system and of the specific applications and Java Virtual Machines in a distributed application. You will want to install agents at every step of the transaction's path, including the Web server, the application server, and the database server.

The PerformaSure Nexus is the central collection point for all of the monitored data. Based on our tests and on Sitraka's documentation, it is a resources-intensive application that is typically installed on its own system. Agents report their timing data to the Nexus, which correlates the events and stores the information in a database. During a load test of a distributed application, the Nexus will be deluged with information, so the processing and memory requirements are pretty high.

Making sense of all the data is the responsibility of the PerformaSure Workstation. The workstation is a GUI application that manages PerformaSure sessions, sets thresholds, and generates reports. It retrieves information from the Nexus and displays it graphically onscreen or in PDF reports.

How It Works

PerformaSure tracks a transaction through a distributed system, gleaning important performance metrics at every step of the way. In a simple Web application, a user visits a Web site and submits a request. That request is forwarded to a middleware application server, which in turn queries data from a database. From the user perspective, this all seems like a single, monolithic application. But behind the scenes, there are any number of systems working together to service the request. A delay in any of these will mean a delay in the user response time. Drilling down and identifying exactly which component is causing the delay is critical to identifying and correcting the bottleneck.

Traditional load and stress tests only identify overall application performance. However, by installing agents at each of these critical services, PerformaSure is able to not only report on overall system performance, but is also able to detect whether a performance problem is due to the network or to the application, to the database back end, or to slow server hardware. Pinpointing the exact source of the slowdown means less time is spent on troubleshooting.

We installed PerformaSure 1.62 on a Windows 2000 environment running BEA WebLogic 6.1. The version we had did not support the latest 7.0 version of WebLogic, which was disappointing. Supported application servers include BEA WebLogic Server 6.0 and 6.1. Web server support includes Apache 1.3.12 and other HTTP servers. (*Note:* PerformaSure v1.7 does support WebLogic Server 7.0.)

In addition to supporting these applications, PerformaSure includes operating system agents for AIX, Solaris, and Windows that track system resource utilization. This is essential for identifying whether or not a performance problem is due to the application or to insufficient or faulty hardware.

Overall, we were impressed with the amount and organization of the information provided by PerformaSure. The workstation software provided us with a wealth of information and troubleshooting control.

We had a love hate relationship with PerformaSure's installation. Installing the agents, Nexus, and workstation is incredibly easy, but configuring them to work correctly is not for the faint of heart. The problem is that getting the environment set up requires the QA engineer to manually modify the configuration files. Manually changing an Apache configuration file or monkeying with the JVM startup script is not our ideal of a smooth installation process. It would be nice if the Nexus, workstation and Agents had their own configuration utilities. Not to mention having to manually edit WebLogic configuration the files. In our case, where one test engineer set up the WebLogic environment and another installed and configured PerformaSure test tool, it became apparent that this is an enterprise test tool that requires an enterprise commitment.

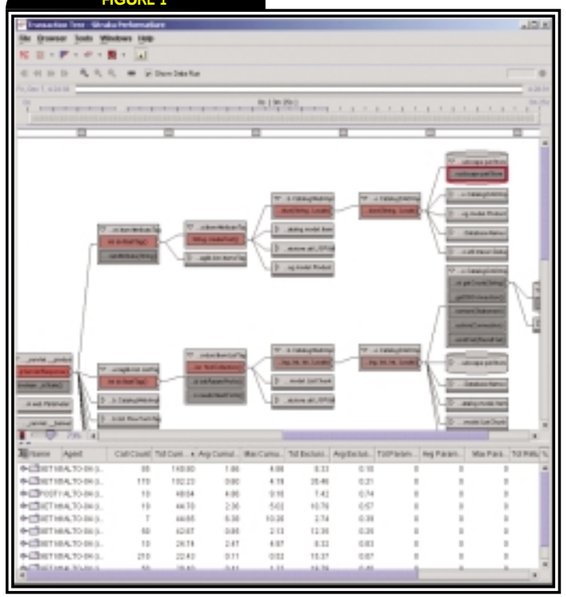
When compared with competing products, such as CompuWare's ServerVantage, it is obvious that there is no simple way to simplify the configuration process. The good news is that once it is configured, everything should run smoothly. Then the bulk of your time will be spent analyzing results.

Summary

Overall, we give PerformaSure high marks in data acquisition and data organization – the two critical areas in performance optimization and troubleshooting. The flexibility in looking at your data and the simplicity of configuring projects will allow QA professionals to pinpoint bottlenecks quickly and easily. We did wish that configuring the PerformaSure software components were more streamlined.

Fortunately, once it's configured you won't have to deal with it. If you're concerned about Web application performance, PerformaSure will give you the information that you need instead of making you wait at the end of the line wondering why your train is late.

FIGURE 1



PerformaSure's Transaction Tree View lets you see which components are causing the bottleneck, tier-by-tier.

AUTHOR BIO...

Ken Brady is a senior test architect at KeyLabs, a full-service software testing and consulting firm. He specializes in developing enterprise IT QA and test methodologies, with a particular focus on security and penetration testing. Ken is a Certified Information System Security Professional (CISSP) with over 10 years of experience in vendor and enterprise software development.

CONTACT...

KeyLabs (www.keylabs.com).

Transactions: the Lingua Franca of Computers...

THE ROUTE TO A QUICK RETURN ON INTEGRATION INVESTMENT

BY PETER HOLDITCH

AUTHOR BIO

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

CONTACT...

Peter.Holditch@bea.com

The waves of IT, as they are often called to, are marked out reasonably accurately by languages. Starting almost at the beginning, take COBOL. With its love of uppercase characters, and overly restrictive attitude to what column the uppercase characters appear in – not to mention its extraordinary zeal for the full stop – COBOL has always struck me as a language for programmers to use to shout at computers. I guess that's a reasonable alternative to feeding them punched cards, or worse, flicking switches on a front panel – you can see why the original COBOL guys wanted to shout!

Since the COBOL era, we have come through various languages, some keener than others on the capital letter, none as controlling about screen columns and full stops (the semicolon has now taken its rightful place as the most feared punctuation mark). During this journey, languages remained relatively free of any notion of distributed processing, and other languages came in to being to fill that gap; for instance, DCE and CORBA's idl, which tried to describe distributed computing systems and subversively surface their interfaces in programming languages in order to try and keep the distribution transparent to the language programmer.

For many reasons (that make a debate in themselves, so please just buy my assertion here) these programming models almost without exception failed to deliver very large scale distributed systems – I believe that was largely because in their effort to hide the mechanics of distribution, designers were lulled into a false sense that any application model could be implemented with any distribution model – leading to many systems which, however beautiful on the drawing board or in the design tool, performed like one legged dead donkeys, spending more time distributing stuff than doing any processing.

The Dawning of a New Age?

So now, we're told, we're at the dawning of a new age – the age of Web services, a new language, based on XML, for the distribution of computing systems. In a bizarre and unexpected throwback to COBOL, Web services messages are pretty text dense – they generally seem to use an alarming quantity of uppercase, and are more rigid about punctuation than you could dream. This time, however, we are witnessing the dramatic rise of the >, <, and \ characters. If COBOL was the language for people to use to shout at computers, Web services has provided a language for computers to use to shout at each other – through Venetian blinds.

However, I contend that this is a good thing. People frown and get paranoid about SOAP messages – they grimace as they think of the bytes passing up and down the network – and this forces them to do what CORBA et al tried to distract them from – They try to minimize the number of monstrous interfaces, they try to send big bulky messages instead of lots of small ones across the interfaces they feel obliged to provide, and they become obsessed with isolating code changes from the punctuation-laden monsters that are flying over their wires. In short, they design good distributed systems. The mantra for good Web service design is asynchronous, coarse grained, loosely coupled, all of which your gut feels you should be doing when you see the things in action.

So, Have I Lost The Plot?

So, have I lost the plot then? You can argue among yourselves about that, but I assert that I haven't. Web services are the new distributed systems approach; transactions are all about providing the infrastructure to enable people to build distributed business systems without having to spend the rest of their lives thinking through and coding for every possible failure mode, so there must be a connection between the two – and there is. So here we go with the column that you thought you were expecting…

To date, most demos of Web services have been rather simple – get a stock quote, get a weather report, etc. That's fine, until you try to implement a real system using them and find that you need the asynchronous behavior that I mentioned before – and start to think through the conse-

quences of any number of potential failures through the distributed system as a whole. You quickly realize that the simple "knock up a servlet with SAX and/or Apache libraries" approach to Web services won't get you very far. As I've said before, transaction coordination is very closely coupled to infrastructure, so if you start messing with XML parsers and JTA APIs to try and build a concept of a distributed business action into your system, then congratulations! You've just won first prize in an infrastructure development beauty contest.

But help is at hand…. From the protocol perspective, BEA, IBM, and Microsoft have come up with a set of specifications to enable the propagation of transaction context between Web services. These two specifications are called WS-Coordination and WS-Transaction. Fundamentally, the thing you need to do to have multiple business operations behave as a cohesive unit is to associate some kind of identifier with all of them. Once you have done this, you can instruct the collection of operations to roll forward or back, without knowing what the operations actually were – so long as you know the correct identifier. This is what WS-Coordination provides – a way to generate this identifier and ship it along with the XML messages that make up the Web service interactions. This identifier is known as a *coordination context*. Provision is also made for identifying a service that can actually provide the coordination services. Note that no mention has been made of what is happening beyond "something is being coordinated" – WS-Coordination defines an abstract coordination relationship. Two concrete implementations of this are defined in the WS-Transaction specification. After all, it is somehow pleasingly symmetrical to have a two-phase standard define a coordination protocol!

Before I launch into WS-Transaction, let's take a step back here… remember that Web services are a way for computers to shout at each other through Venetian blinds. This isn't the kind of thing that you want a lot of going on, we already agreed on that when thinking about how to implement distributed systems. Imagine an atomic transaction…. All those pre-commit, prepare, and commit messages flying back and forth.

If you're not feeling queasy by now, you're clearly not imagining hard enough. Also, remember the Web services design mantra… the word "asynchronous" appeared there, didn't it? So did the phrase "loosely coupled." A conventional two-phase transaction is a great way to achieve synchronous coupling between things. We already decided that this is to be avoided, so what gives? Read on!

It won't surprise you to know that the WS-Transaction spec provides ways to minimize the impact of cohesive behavior on the performance of the system. First, there is explicit support for interposed managers. Imagine that your transaction coordinator was on the other side of the world, and all those prepares, commits, and so on were flying over a slow WAN. In the case where the service you're making available locally is composed of a number of other Web services, you don't want the internal workings exposed across the WAN and coordinated on that basis – after all, that would be fine grained. The spec allows the local end of the connection to interpose its own transaction manager so that the remote side sees a single resource to prepare and commit – the fact that it may map to many coordinated resources on the local side, each of which will need its own flurry of prepares and commits, is taken care of by the local interposed coordinator, achieving the coarse granularity that we seek.

> From the protocol perspective, BEA, IBM, and Microsoft have come up with a set of specifications to enable the propagation of transaction context between Web services

So, I hear you cry, what about asynchronicity? Well, that's taken care of too. The WS-Transaction specification provides for two *coordination types*, called AT and BA standing for atomic transaction and business activity.

An atomic transaction is what you expect, know, and love… a synchronized set of related prepares and commits to being all the work in an identified unit to an ACID close. A business activity, on the other hand, is expected to be long lasting – maybe months, the term of a contract perhaps. In a business activity, individual changes are made permanent and visible as they happen, with the ability to back out changes (based on their coordination context), if the need arises some time in the future, with what is known as a compensating action – for example, cancelling a reservation, refunding a payment, etc. This isn't trying to make it look as if the activities didn't occur, but rather to take some new action which ensures that every party in the activity reaches a new, mutually acceptable state.

It's Time to Run!

So, the WS-Coordination and WS-Transaction specifications together identify how systems should communicate in order to get reliable, business-meaningful results from a Web service–based system. This brings the goal of using Web services as the foundation for a standards based integration solution a big step closer.

Back on the developer's view, however, the other half of the problem is implementing the code that maps this protocol onto some underlying transaction manager. It is here that the development cost will mount up. It is also here that technologies like BEA's WebLogic Workshop can be expected to come to the rescue – transaction coordination is another infrastructure-level requirement that will be best delivered by a runtime engine such as that which underpins the Workshop system.

Then, finally, truly useful transactional behavior will be delivered to the masses, and quick return on integration investment will surely follow. Perhaps shouting through window dressings isn't so stupid after all.

This is the third and final article in a series aimed at demystifying the frameworks that compose the BEA WebLogic Portal.

Understanding the Portal Framework

AGGREGATION OF VISUAL PORTAL COMPONENTS

BY **DWIGHT MAMANTEO**

Although this month I focus on the portal framework, you should realize that the value of WebLogic Portal is in the combination of frameworks, components, and wizards that enable the creation of Web-based applications that provide value to the end customer and the hosting business. The customer benefits by being presented with a solution that is easy to use, personalizes content and services, and provides a much better user experience. The business saves money by maintaining only one portal solution that provides a view into all of its back-end systems, requiring only one portal solution throughout its departments, having a decoupled solution that allows the changing of back-end systems without enforced change to the front-end enterprise portal, and reusing application components throughout the enterprise. The core business also improves by having a solution that intelligently adapts itself to the user's dynamic user profile, provides the ability to intelligently execute campaigns, and provides the means to dynamically modify the application logic after the business analysts have completed their analysis of their online business, thus providing a 360-degree feedback loop.

The portal framework is an adaptable presentation engine that aggregates visual presentation components, called portlets, into a single presentation unit and enables the interaction between these presentation units to create a dynamic and interactive Web front end to a business application. The main components in the portal framework are the Flow Controller, Portal Page Generator, Portlet Page Generator, Portlets, and custom JSP tags. This article will describe each component in detail and walk the reader through a framework execution scenario.

AUTHOR BIO...

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

CONTACT...

dwight@bea.com

Portal Framework

The portal framework can be logically divided into two major functional areas. The first is the aggregation of visual components, and the second is the management of the dynamic interaction between the different visual components and the management of incoming requests from the user. In the interest of not turning this article into a voluminous white paper, I will focus only on the aggregation of visual portal components. Those of you who want a more detailed understanding of the interaction between the different portlets, the management of incoming requests, and the wizards that increase developer productivity, are encouraged to read the material located on BEA's online documentation Web site.

The main components of the portal engine responsible for aggregating the visual components in the portal framework are the PortalWebflowServlet, the portal.jsp page, the portlet.jsp page, and the custom JSP tags (see Figure 1). The PortalWebflowServlet uses the same architecture as the WebflowServlet architecture described in "Building Adaptive Applications with Reusable Components" (***WLDJ***, Vol. 1, issue 6), and is the cornerstone for handling the inter-portlet communication and the management of incoming requests (see Figure 2).

Flow Controller

The PortalWebflowServlet is the Java servlet responsible for handling flow control for the portal framework. This portal flow controller intercepts incoming requests from the client and determines whether the request is intended for a portal Web application. If the request is intended for a portal Web application, the servlet hands control to the portal Webflow executor, which would invoke the portal page generator (see Figures 1 and 3). The portal page generator then manages the creation of the portal page and all of its composite parts.

Portal Page Generator

The portal.jsp file is the portal page generator responsible for combining the disparate visual components that make up the portal page. The portal page generator would aggregate the header and footer content for the page, and incorporate the content returned by the portlet page generator (see Figure 3). The functionality contained in the portal.jsp page can be extended by the portal developer to include custom functionality. However, it is not recommended to hard-code custom functionality into the page, as there is no guarantee that the portal page generator will not change in future releases of the product.

Portlet Page Generator

The portlet.jsp file is the portlet page generator responsible for combining all the visual portlet components into a compound visual portlet unit. The portlet page generator would aggregate all the different portlets into an HTML table that conforms to the layout and visibility strategies determined by the portal administrator. The availability of portlets is governed by the business rules processed by the entitlements engine, which also makes use of the user's user and group profile information. Like the portal page generator, the functionality contained in the portlet.jsp page can also be extended by the portal developer to include custom functionality. Again, it is not recommended that you hard-code custom functionality into the page, as there is no guarantee that the portlet page generator will not change in future releases of the product.

Portlet Pages

Portlet pages are visual components that provide specific business functionality and content. Each of the portlet pages or components can be considered to be a single reusable visual unit that is shared amongst different portal deployments. A portlet can be constructed to provide services or data from one single back-end source or multiple back-end sources. Each portlet component can be composed of a combination of the following subcomponents: titlebar, banner, header, footer, and content section. Some of the out-of-the-box portlets that come with the product include functionality for collaboration, calendaring, Webmail, chat, whiteboard, and discussion groups.

Portal and Portlet Management JSP Tag Library

These custom management tags allow JSP developers to access the portal framework functionality of WebLogic Portal without having to write EJB client code. The portal framework management JSP tag libraries are divided into three main categories; portal, portlet and utility. The JSP: tags in the portal category are used to execute management functionality that is specific to the internal workings of the portal as a whole; the JSP tags in the portlet category are used to execute portlet specific functionality; and the JSP tags in the utility category are used to perform URL validation. (For more information on the parameters for each JSP tag and for the entire WebLogic Portal JSP tag library, visit the BEA online documentation Web site: http://edocs.bea.com.)

Portlet JSP Tags

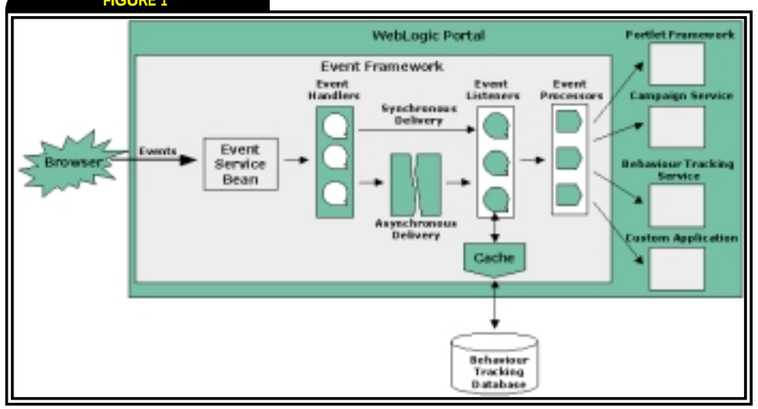
Instead of having the portlet developer write custom code to call the PortalWebflowServlet, custom JSP tags are provided to increase productivity and to minimize the potential of malformed calls. Portlet JSP tags are provided to properly construct a Webflow URL, an HTML form, and an HTML form that takes advantage of the data validation component in the portal framework. Additionally, custom JSP tags are provided to display an Edit portlet, a minimized portlet, an unminimized portlet, a maximized portlet, an unmaximized portlet, a floating portlet, and the exceptions thrown by the Webflow processor. Each of the tags in the portlet JSP tag group is focused on managing each portlet component and is meant to be called in the portlet content page.

More information on the portlet JSP tags can be found at http://edocs.bea.com/wlp/docs70/jsp/mngprtal.htm#1002046.

Portal JSP Tags

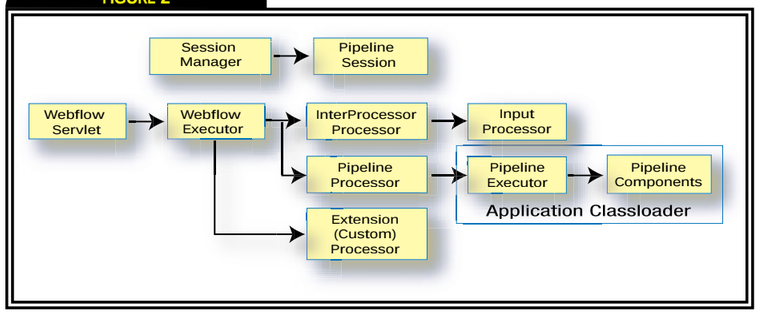
Like the portlet JSP tags, the portal JSP tags are used mainly to aid the portal developer by providing helper tags that increase

FIGURE 1



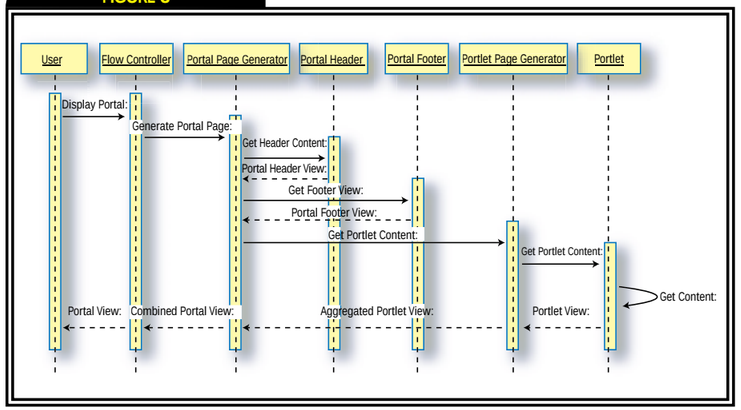
Portal Visual Aggregation Framework

FIGURE 2



Webflow Framework

FIGURE 3



High-level scenario

the developer's productivity. Portal JSP tags are provided to properly construct a Webflow URL, an HTML form, and an HTML form that takes advantage of the validation component in the portal framework. Additionally, a custom JSP tag is provided to notify the PortalWebflowServlet of the event that the user is changing a page. Each of the tags in the portal JSP tag group is focused on managing the portal solution as a whole and is meant to be called in the portal page generator file.

More information on the portal JSP tags can be found at http://edocs.bea.com/wlp/docs70/jsp/mngprtal.htm#1003563.

Utility JSP Tags

The two utility JSP tags provided are used as sanity processing rules in the JSP tag.

Coupling the portal framework with the personalization, entitlements, and campaign frameworks provides the ability to create intelligent applications that can adapt themselves to each specific user

The first utility JSP tag will process the code contained with the brackets if the URL provided is valid. The second utility JSP tag will process the code contained within the brackets if the URL provided is invalid. Since these tags can be used in either the portal page or any of the portlet pages, they are part of the utility category.

More information on the utility JSP tags for the portal framework can be found at http://edocs.bea.com/wlp/docs70/jsp/mngprtal.htm#1004198.

Conclusion

The portal framework enables the reuse of visual components, and manages the interaction between the visual components and any incoming requests. Coupling the portal framework with the personalization, entitlements, and campaign frameworks provides the ability to create intelligent applications that can adapt themselves to each specific user.

My goal in this series of articles was to provide a clearer understanding as to the components and functionality that the portal, event, and advisor frameworks provided. The key concepts that are embodied in each of these frameworks are that they can be used in a unified manner, they are simple to use, and they can easily be extended to suit custom business needs.

Finally, I strongly suggest that each of you visit the BEA Portal Solution Center (http://portalsolutions.bea.com/index.jsp). This Web site contains white papers; development documents; sample code; and a catalog of partner portlets, integrated offerings, and total portal solutions.

WIRELESS

– continued from page 10

whose transactions are container-managed.

Notice that as in solution 1, you still need to manage your own cache sessions (containing EJB headless rather than HTTP cookies).

This can be implemented in a variety of ways. The simplest is to make it a database cache. If performance is an issue, you can purchase a third-party hardware cache box which can give you better performance, since these tend to be memory-based caches. The Java Community Process is addressing caching through its jcache component; this will certainly fill a void in J2EE if and when it becomes standard.

Accessing State Locally

One of the keys to this design is that, as with servlets, request processing is sent to the data rather than vice-versa. The goal is to access session state locally. To achieve this, the SFSB mediates all communications between the client (protocol adapter) and the application (a collection of EJBs replicated in the cluster). Each coarse-grain service offered by the application can only be accessed by a Java object whose sole purpose is to call the service on behalf of the client. This object is called the service executor and employs a "Strategy design pattern." The client creates an instance of the service-specific executor and passes it to the SFSB. In turn, the SFSB invokes a generic method on the executor which ultimately calls the back-end service. Using this framework guarantees that the request made from the protocol adapter will be processed on the host that contains the session's state.

Conclusion

Whichever solution you choose, much can be leveraged from WebLogic and J2EE. This can make a big difference in delivering your product faster using dependable, out-of-the-box components.

Resources

- Gridley, Michael; Woolen, Rob; Emerson, Sandra L. (2002). *J2EE Applications and BEA Weblogic Server.* Prentice-Hall.
- Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.
- *Failover and replication in a cluster:* http://e-docs.bea.com/wls/docs70/cluster/failover.html
- ***Wireless Village and the IMPS initiative:*** www.wireless-village.org
- *The SMS forum:* http://smsforum.net

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altaworks	www.altaworks.com	603-598-2582	52
Altova	www.altova.com	978-816-1600	11
BEA Systems	http://dev2dev.bea.com/useworkshop	800-817-4BEA	3
Borland	www.borland.com	831-431-1000	17
Dirig Software	www.dirig.com/illusion/weblogic	603-889-2777	9
EnginData Research	www.engindata.com		45
Hewlett-Packard	www.hp.com/go/developers	650-857-1501	23
Intel	www.intel.com/ad/bea	408-765-8080	19
KeyLabs	www.keylabs.com	888-KEY-LABS	28
ObjectFocus	www.objectfocus.com	408-973-8200	47
PANACYA	www.panacya.com	877-726-2292 x3344	51
Performant Software Solutions	www.performant.com/weblogic1	866-773-6268	29
PowerBuilder Developer's Journal	www.sys-con.com/pbdj	888-303-5282	36
PwC Consulting	www.pwcconsulting.com/nwn	267-330-6355	25
Sitraka	www.sitraka.com/jclass/wldj	800-663-4723	2
Sitraka	www.sitraka.com/performance/wldj	800-663-4723	13
SYS-CON Media	http://developer.sys-con.com	201-802-3020	27
SYS-CON Publications	www.sys-con.com	201-802-3020	30, 37, 45, 48-9
SYS-CON Reprints	www.sys-con.com	201-802-3026	41
Web Services Edge 2003	www.sys-con.com	201-802-3069	31
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	35
Wily Technology	www.wilytech.com	888-GET-WILY	4

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

A Guided Tour of the WebLogic E-Business Platform

BY **ALI AKBAR, KEYUR SHAH**

Those who understand e-business know that the Internet has forever changed the business world. More and more successful companies are harnessing and exploiting the power of the Internet to maintain a competitive advantage.

Although e-business offers great opportunities, it also demands fundamental changes in the way business is conducted, especially in terms of the following:

- Using the Internet for advertising, sales, and customer support
- Presenting a new form of direct customer relations
- Providing new roles for intermediaries
- Implementing major changes in products and services

Building e-business using the latest tools and technologies and integrating them with existing applications is a major challenge faced by most businesses. There is a demand for tools and products to help bridge the new world of Web-based solutions to the ancient mainframe applications.

This chapter provides a brief introduction to various e-business products available from BEA, such as eLink, WebLogic Portal with personalization and commerce services, WebLogic TUXEDO, and WebLogic Java Adapter for Mainframe.

WebLogic eLink

To understand this product, you should know a thing or two about Enterprise Application Integration (EAI). EAI products leverage enterprise application transaction platforms to integrate existing legacy applications with customer-focused (business-to-customer, or B2C) and e-commerce (business-to-business, or B2B) initiatives. Typically, organizations have already been running legacy applications for decades. In response to the Web revolution and the availability of Web-based applications to deliver services to clients, they might need to integrate back-end legacy systems with new advances to perform rapid requests for information. EAI makes that possible. WebLogic eLink is just one of BEA's seamless solutions for integrating legacy systems with new Web-based applications.

The BEA eLink family offers two solutions for these integration needs: eLink Platform and eLink Foundation (see Figure 1). The eLink Platform provides support for eLink adapters to customers with lower EAI requirements. BEA adapters provide access to external enterprise applications, such as Customer Resource Management (CRM, including Vantive, Siebel, and Clarify); Enterprise Resource Planning (ERP, including SAP R/3, PeopleSoft, and Oracle); mainframe applications (including CICS, IMS, any SNA peer, Unisys OS 2200, Unisys MCP/AS, or any system that supports Open Group XATMI over OSI TP); and billing applications, including Portal. eLink Foundation is for customers who have more complex EAI scenarios to handle.

WebLogic Express

WebLogic Express is a WebLogic Server product with limited facilities, as compared to the full-blown WebLogic Server software. It can be used to host Web pages; to implement Java applications such as applets, servlets, and Java Server Pages (JSPs); to generate contents dynamically; and to let applications communicate with databases using the Java Database Connectivity (JDBC) implementation. It also provides Remote Method Invocation (RMI) functionality. Unlike WebLogic Server, WebLogic Express does not support EJBs, Java Messaging Service (JMS), and the two-phase commit protocol for the transactions. WebLogic Express is a scalable platform that serves dynamic contents and data to Web applications and wireless applications. It provides a Web container but not a Java 2 Enterprise Edition (J2EE) container.

BEA WebLogic Portal 4.0

Prior to WebLogic Portal 4.0, separate tools were used for portal development, commerce services, personalization services, and campaign management. Now WebLogic Portal puts all of these Web applications in one package. This makes for a better product with rock-solid integration between functionalities and services. Figure 2 shows Portal server facilities.

Portal Services

Portal is a term used to describe a World Wide Web site proposed to be a major starting site for users, or one that users tend to visit as an anchor site. Portals are categorized as general portals and *specialized* or niche portals. Some major portals include Yahoo!, Excite, Netscape, CNET, Microsoft Network, and America Online. Niche portals include *Garden.com* (for gardeners), *Fool.com* (for investors), and *SearchNetworking.com* (for network administrators).

A portal can also be considered a network service that provides access to various heterogeneous network services that are local and remote, structured and unstructured. Such network services might typically include resource discovery services, e-mail access, and online discussion forums.

Portals such as MSN, shown in Figure 3, make use of contents provided by their partners, and they keep users on their sites by providing content and services.

Portals typically include special functionality called *resource discovery*; the user can potentially search for various information or products, such as travel information, weather information, latest music albums, and so on. Figure 4 shows some examples of resource discovery interfaces.

WebLogic Portal is built on the WebLogic E-Business platform and is designed to help lower the cost involved in accessing the information, applications, and business processes for enterprises, customers, and business partners. WebLogic Portal simplifies the creation of personalized and sophisticated portal sites by providing a user interface framework, prebuilt presentation elements, and templates.

E-Commerce Services

One of the most popular Internet myths is that running an online store is simple: all the customer has to do is point, click, and buy. However, the reality is that carrying out a successful e-commerce venture is far more complex than developing and maintaining another type of Web site.

You must consider many elements when developing an online store. Your Web site needs to provide customers with information about your products. A payment-processing system must be in place—not to mention a way to calculate taxes properly—and all of this on top of providing customers with the kind of personalized service that will keep them coming back. This is where BEA WebLogic Portal 4.0 with e-commerce services can come into play.

WebLogic Portal 4.0 provides a built-in set of commerce templates that developers can use to build e-commerce Web sites easily and effectively, including a products catalog, order processing system, tax calculation method, payment processing structure, and registration.

Personalization Services

By personalization, we mean that the individuals involved in Web content development have the capacity to tailor an application to a particular individual or a group of individuals in an organization. The customization depends on predefined user attributes, such as age or gender. In this scenario, individual customers can set up their own profiles, and the content delivery to the individuals accessing the Web site will depend on each user's particular profile. The core benefit of personalized services is that a complete solution can be developed rapidly that allows your business to extend its competitive advantage, accelerate response time to customers, and meet market demands. WebLogic Portal 4.0 provides personalized services that can be used with JSP tags to deliver a responsive, customized experience to the users or group.

Campaign Management

Internet marketing is where organizations are currently investing most of their dollars, both in releasing their own ad campaigns and in working with popular sited hosts to carry it out for them. Almost all Web sites host and carry out ad campaigns for their partners on the Web.

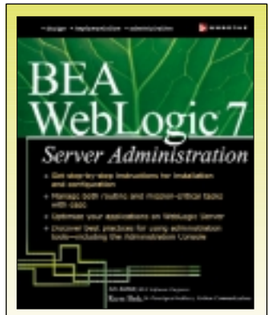
FIGURE 1

```
C:\WINNT\System32\cmd.exe

C:\>java weblogic.Deployer -adminurl http://localhost:8001 -username  installadm
inistrator  -password installadministrator -verbose  -source C:\test\exploded\ex
ploded_war -name war -targets myserver -activate
Operation started, waiting for notifications...
[Id: 1] preparing war for myserver
[Id: 1] prepared war for myserver
[Id: 1] activating war for myserver
[Id: 1] activated war for myserver

#TaskID Action          Status  Target          Type    Application     Source
1       Activate        Success myserver        Server  war     C:\test\exploded
\exploded_war
C:\>
```

eLink architecture

FIGURE 2

Portal server facilities



ISBN 0-07-222316-2

Publication Date:
10/21/2002

Pages:
432

Price:
$49.99

Available at
www.osborne.com

WebLogic Java Adapter for Mainframe

Most large organizations have ancient mainframe applications that serve their business needs, but they also have multi-tier, Web-based applications that serve the needs of their business in a modern way. The success of organization depends on how these two are integrated. BEA WebLogic Java Adapter for Mainframe (JAM) provides a solution that delivers bidirectional and request-response integration between Java and mainframe applications. In fact, JAM extends the life of mainframe applications by integrating them with Java applications running on WebLogic Server. Bidirectional means that once a WebLogic JAM is installed, it allows the WebLogic Server application to invoke mainframe applications. A typical WebLogic JAM distributed configuration is demonstrated in Figure 5.

The obvious benefits of using BEA WebLogic JAM, from business perspectives, includes leveraging existing IT investment; having an easy-to-use, Web-based Administration Console for administering JAM and providing support for a two-phase commit for distributed transactions; generating J2EE applications from existing mainframe applications; providing data translation among Java, XML, and COBOL data types; supporting mainframe security credentials; and integrating JAM into a WebLogic Integration.

WebLogic JAM and mainframe systems have two main components for interaction: gateway and Communications Resource Manager (CRM). Figure 6 demonstrates the WebLogic Administration Console for administering WebLogic JAM.

WebLogic JAM supports the following dynamic configuration changes:

- Adding and removing WebLogic JAM gateways
- Enabling and disabling WebLogic JAM gateways
- Adding and removing links
- Enabling and disabling links
- Adding and removing services
- Enabling and disabling services

eGen Application Generator

BEA WebLogic JAM provides a utility called eGen that assists in quickly integrating a mainframe application. The command-line utility generates Java source code using a COBOL copybook and a declarative script file as input. On executing the utility with necessary input, it generates a working Java skeleton application that can be used as a basis for new applications or as an addition to an existing application. The eGen utility can generate servlets, EJBs, or stand-alone clients. All generated skeletons include code that implements connectivity and data translation to mainframe applications.

With the help of the *eGen* utility, you can generate four different types of Java application models: client class, client EJB, server EJB, and servlets.

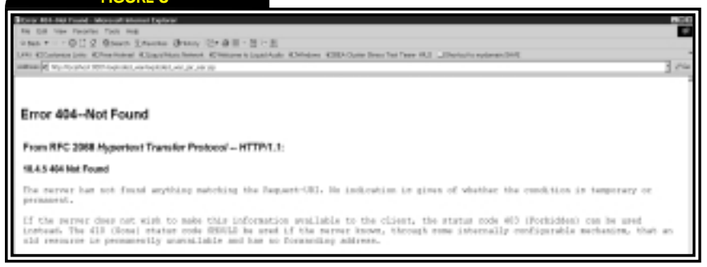
The client class is a stand-alone Java class file that invokes mainframe services. It may be incorporated into your own EJB or utilized in some other way within your code.

The client EJB is a stateless session EJB that invokes mainframe services. A servlet or other client programs can call it. This is the normal model for building a production application with access to mainframe services. A servlet that invokes the EJB's methods may be added for testing or demonstration purposes.

The server EJB is a stateless session EJB that provides a service to mainframe client applications.

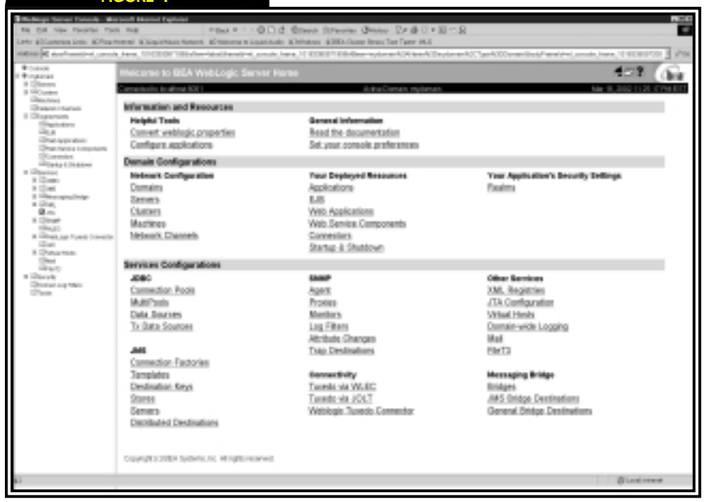
The servlet-only application is a servlet that presents a simple form and invokes mainframe services directly. This model is useful for testing purposes but is not suitable for production applications.

FIGURE 3



MSN portal site

FIGURE 4



Resource discovery

As a WebLogic Server administrator, you will be required to install a gateway and make modifications to the WebLogic configuration to start it automatically with WebLogic Server. Follow these steps:

1. Install the WebLogic JAM gateway.
2. Deploy WebLogic JAM in the WebLogic Server environment.
3. Run the IMS (Information Management System) installation verification sample.
4. Run the CICS (Customer Information Control System) installation verification sample.
5. Define a WebLogic JAM gateway.
6. Configure mainframe client applications.
7. Expose enterprise Java Beans to the mainframe.
8. Expose JMS events to the mainframe.
9. Expose WebLogic integration events to the mainframe.
10. Start and stop a gateway.
11. Activate and deactivate CRM links.
12. Monitor the CRM status.
13. Monitor CRM trace-level settings.
14. Monitor CRM links.
15. Enable an exported WebLogic application.
16. Enable a service.

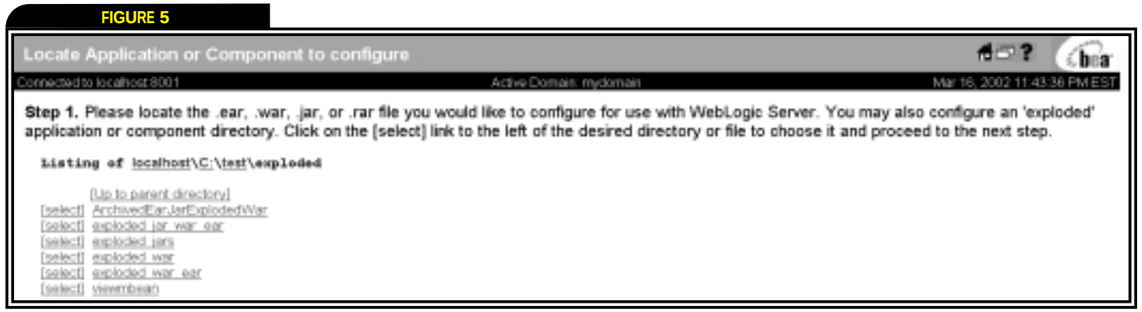
NOTE: WebLogic JAM configuration information is persisted on the administration node of the WebLogic domain. WebLogic JAM configuration is stored in Extensible Markup Language (XML) in a single file, jamconfig.xml, which should reside in the domain-specific directory on which WebLogic JAM runs.

WebLogic and TUXEDO

A robust transaction processing system, TUXEDO (Transactions for UniX, Enhanced for Distributed Operations) is the following:

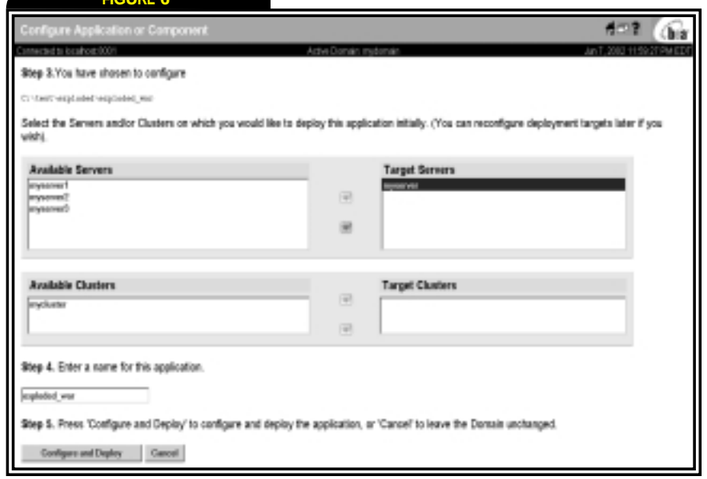
- A middleware: It relays requests and responses between client and server processes (with or without transactions).
- A transaction processing (TP) monitor: It begins, terminates, and monitors transactions on behalf of client and server processes.
- A distributed TP monitor (DTP): It allows transaction participants to be located on different machines or associated with different databases.

FIGURE 5



WebLogic JAM distributed configuration

FIGURE 6



WebLogic JAM distributed configuration

Those who are tired of using low-level networking (streams, sockets, daemons, and so forth) for transaction processing requirements can use TUXEDO as their software communication infrastructure.

BEA TUXEDO is a software bundle that allows integrators and developers to rapidly develop robust client-server applications. It provides an interface called Application to Transaction Monitor Interface (ATMI) that implements most of the code necessary for clients and servers to communicate. Application programmers divide business logic among TUXEDO-controlled programs or processes called servers. Other programs, called clients, can independently call functions that implement specific tasks within these servers. These tasks are called *services*. The following are services rendered by TUXEDO:

- Implements a business task.
- Encapsulates data access.
- Provides location transparency by helping clients call services by name and not by server.
- The TUXEDO bulletin board provides name resolution.

TUXEDO provides a runtime graphical administration tool to configure and monitor applications. Its applications can span multiple machines, even those with different internal architectures. Logical applications or domains can be configured to facilitate interapplication communications.

NOTE: BEA TUXEDO is a middleware infrastructure for enabling distributed transaction processing and building scaleable three-tier client/server applications and heterogeneous environments.

TUXEDO Servers and Clients

TUXEDO servers provide the following:

- Business logic by TUXEDO services
- Several encapsulated services
- Resource management
- Not visible for clients

Following is a list of potential candidates for TUXEDO:

- Part of the front-end applications
- Presentation layer for user input/output
- Senders of application business logic data

Checklist

In this chapter, we made a straightforward journey into the world of e-business products and the new innovations that that BEA is making in this arena. We covered the following:

- WebLogic eLink is used for EAI.
- WebLogic Express is a limited version providing restricted functionality of WebLogic Server.
- WebLogic Portal is a portal development tool leveraging commerce services, personalization services, and campaign management.
- WebLogic Java Adapter for Mainframe is used for integrating WebLogic Server with back-end systems.
- eGen is used as an application-generation tool.

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES ONLINE

Go To www.sys-con.com/suboffer.cfm

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

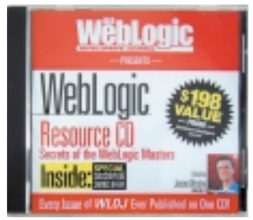
RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

and receive your **FREE** CD Gift Package VIA Priority Mail

Each CD is an invaluable developer resource packed with important articles and useful source code!

More than 1,400 Web services and Java articles on one CD! Edited by well-known editors-in-chief Sean Rhody and Alan Williamson, these articles are organized into more than 50 chapters on UDDI, distributed computing, e-business, applets, SOAP, and many other topics. Plus, editorials, interviews, tips and techniques!
LIST PRICE $198

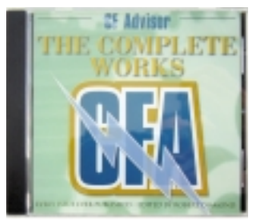
The most complete library of exclusive *JDJ* articles compiled on one CD! Assembled by *JDJ* Editor-in-Chief Alan Williamson, more than 1,400 exclusive articles are organized into over 50 chapters, including fundamentals, applets, advanced Java topics, Swing, security, wireless Java,... and much more!
LIST PRICE $198



The most complete library of exclusive *WLDJ* articles ever assembled! More than 200 articles provide invaluable information on "everything WebLogic", including WebLogic Server, WebLogic Portal, WebLogic Platform, WebLogic Workshop, Web services, security, migration, integration, performance, training...
LIST PRICE $198

The most complete library of exclusive *CFDJ* articles on one CD! This CD, edited by *CFDJ* editor-in-chief Robert Diamond, is organized into more than 30 chapters with more than 400 exclusive articles on CF applications, custom tags, database, e-commerce, Spectra, enterprise CF, error handling, WDDX... and more!
LIST PRICE $198

The largest and most complete library of exclusive *XML-Journal* articles compiled on one CD! Edited by well-known editors-in-chief Ajit Sagar and John Evdemon, these articles are organized into more than 30 chapters containing more than 1,150 articles on Java & XML, XML & XSLT, <e-BizML>, data transition... and more!
LIST PRICE $198



The complete works of *CF Advisor* are now on one CD! Check out over 200 exclusive articles on topics such as e-commerce, custom tags, Fusebox, databases, object-oriented CF, WDS, upgrading CF, wireless, Verity, and more. Plus, find interviews, editorials, and source code!
LIST PRICE $198

Pick the CDs to go with your Multi-Pack order

- Pick one CD with your 3-Pack order
- Pick two CDs with your 6-Pack order
- Pick three CDs with your 9-Pack order

- ☐ Web Services Resource CD
- ☐ Java Resource CD
- ☐ WebLogic Resource CD
- ☐ ColdFusion Resource CD
- ☐ XML Resource CD
- ☐ CF Advisor Complete Works CD

Your order will be processed the same day!

Pick a 3-Pack, a 6-Pack or a 9-Pack ☑

TO ORDER:

☐ 3-Pack	☐1YR ☐2YR	☐ US	☐ Can/Mex	☐ Intl.	
☐ 6-Pack	☐1YR ☐2YR	☐ US	☐ Can/Mex	☐ Intl.	
☐ 9-Pack	☐1YR ☐2YR	☐ US	☐ Can/Mex	☐ Intl.	

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

☐ **Java Developer's Journal**

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

☐ **Web Services Journal**

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

☐ **.NET Developer's Journal**

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

☐ **XML-Journal**

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

☐ **WebLogic Developer's Journal**

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

☐ **ColdFusion Developer's Journal**

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

☐ **Wireless Business & Technology**

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

☐ **PowerBuilder Developer's Journal**

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

3-Pack

Pick any 3 of our magazines and save up to $275.00

Pay only $175 for a 1 year subscription plus a **FREE CD**

- 2 Year – $299.00
- Can/Mex – $245.00
- All Other Cnty. – $315.00

6-Pack

Pick any 6 of our magazines and save up to $350.00

Pay only $395 for a 1 year subscription plus **2 FREE CDs**

- 2 Year – $669.00
- Can/Mex – $555.00
- All Other Cnty. – $710.00

9-Pack

Pick all 9 of our magazines and save up to $400.00

Pay only $400 for a 1 year subscription plus **3 FREE CDs**

- 2 Year – $839.00
- Can/Mex – $695.00
- All Other Cnty. – $890.00

News & Developments

New Software Brings Answers to Complex Business Questions

(San Jose, CA) – BEA Systems, Inc., the world's leading application infrastructure software company, has announced the general availability of BEA Liquid Data for WebLogic, a product that makes it possible to get fast answers to business questions that were prohibitively difficult to answer before. BEA Liquid Data for WebLogic provides integrated views of enterprise information from virtually any data source: databases, XML files, Web applications, integration adapters, and more. These logical views are reusable and can quickly aggregate the latest real-time information from both inside and outside large organizations.

BEA Liquid Data for WebLogic is available as an add-on to the BEA WebLogic Enterprise Platform.
www.bea.com

BEA and Sun Team to Deliver Web-Based Training

(San Jose, CA and Santa Clara, CA) – BEA Systems, Inc., and Sun Microsystems, Inc., a leading provider of hardware, software, and services for network computing, have agreed to broaden their global relationship and offer new learning solutions to customers. The two companies will provide Web-based training on the BEA WebLogic Workshop development environment, as well as on Java technologies, XML, Linux, and the Solaris Operating Environment.

BEA and Sun will offer courses that customers can take at their own pace, virtually anytime, from the comfort of their Web browsers. For more information or to register for these new offerings, visit http://education.bea.com, www.bea.com, www.sun.com

BEA WebLogic Workshop More Productive than WebSphere

(San Jose, CA) – BEA Systems, Inc., the world's leading application infrastructure software company, has revealed more evidence that BEA WebLogic Workshop is seven times more efficient for developing complex Web services than IBM WebSphere Application Developer, based on the number of lines of code required to build the Web service in each environment. This provides further evidence that the productivity gap widens as application complexity increases. As a result, BEA has once again challenged IBM to provide even half of BEA's productivity to developers.

BEA has demonstrated on its dev2dev Web site how developers can build a sophisticated Web service with WebLogic Workshop. The Web service was created with only 26 lines of simple code, in contrast with the 185 lines required to replicate this Web service using IBM WebSphere Application Developer. Even with more lines of code, the IBM-built application fails to match the key reliability features provided by BEA. The demonstration and a comparative analysis can be found at http://dev2dev.bea.com/products/product.jsp?highlight=wlw .

Rapidly Growing SYS-CON Media Garners Awards

(Montvale, NJ) – SYS-CON Media has been named one of the fastest growing 500 technology companies in North America by Deloitte & Touche in its 2002 Technology Fast 500. The announcement came one week after SYS-CON was named one of the nation's fastest-growing private companies by Inc 500 for the third time.

SYS-CON Media is widely recognized in the *i*-technology and magazine publishing industries as the world's leading publisher of print magazines, electronic newsletters, and accompanying Web portals. The company has solidified its dominant role in the *i*-technology space with the 2000 launch of an events business with trade shows, conferences, and education seminars.

SYS-CON Media has achieved a record 752% growth in the past five years. The company's revenue and earnings have grown dramatically since its inception in 1994. In 2003, the company projects its gross margin to increase 51.9%, and the contribution is projected to increase 70.4%, keeping SYS-CON in an impressive growth pattern for 2002, 2003, and beyond.
www.sys-con.com

Precise Automates Application Performance Management

(Westwood, MA) – Precise Software Solutions, a leader in optimizing customers' business through application performance management, has introduced an industry breakthrough that substantially enhances customer productivity by reducing application performance problem resolution time. Precise i3 version 6.0 combines in-depth data collection technology with its TotalCorrelation and SmarTune technologies to resolve application performance slowdowns before they damage business performance. Problem resolution time is reduced to just minutes, compared to the 26-hour average time.

In addition, Precise i3 version 6.0 provides seamless integration between solution components, in-context navigation from tier to tier, enhanced speed of installation and implementation, and application-specific enhancements. www.precise.com

CocoBase Enterprise O/R 4.0 Integrates with WLS 7.0

(San Francisco) – THOUGHT Inc., has announced the integration of BEA Weblogic Application Server version 7.0 with CocoBase Enterprise O/R version 4, Object to Relational Mapping tool. CocoBase's revolutionary Dynamic Object to Relational Mapping architecture simplifies the task of persisting relational data in Java-based applications.

CocoBase provides a standards-based and open and integrated solution to the developer. Once the data is mapped, developers have a full range of choices for persisting their applications on the BEA WebLogic Application Server. Their integration with WebLogic and the leading Java case-tools enables developers to use their tools of choice for an all-in-one engineering solution.
www.thought.com