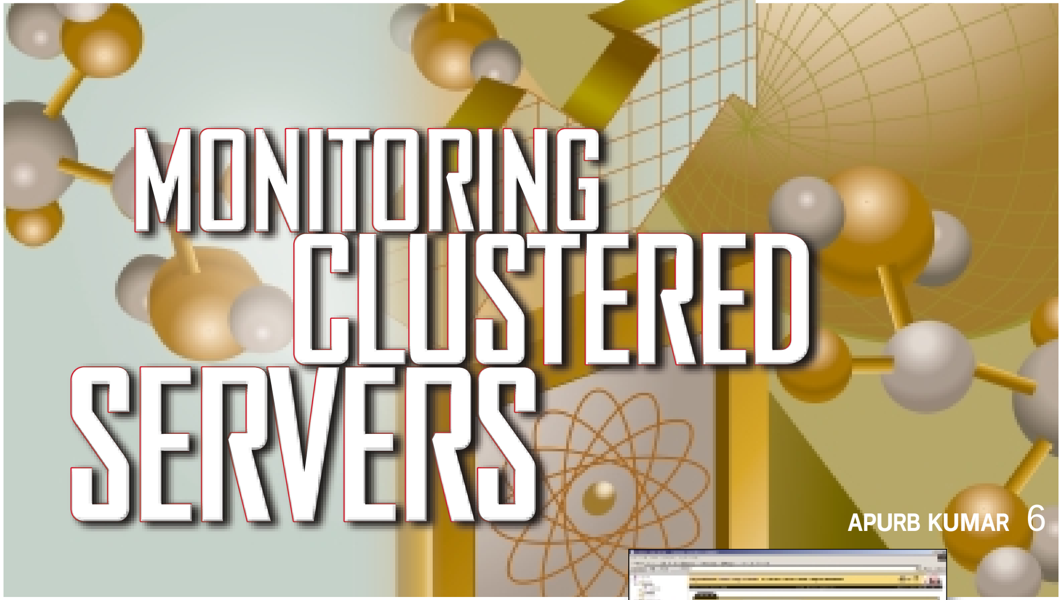
WHERE IS THE WORLD WIDE WEB GOING?

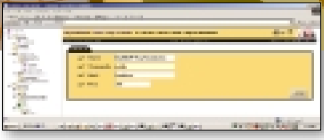
BEA WebLogic DEVELOPER'S JOURNAL

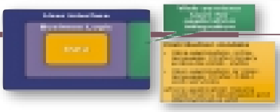
MAY 2003 - Volume:2 Issue:5

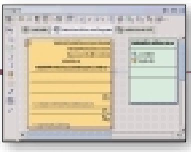
weblogicdevelopersjournal.com

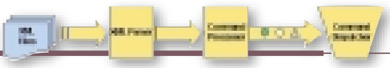


MONITORING CLUSTERED SERVERS









RETAILERS PLEASE DISPLAY UNTIL JUNE 30, 2003
$8.99US $9.99CAN

SYS-CON MEDIA

ReportingEngines

www.reportingengines.com/download/wldj1.jsp

Quest Software

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

Wily Technology

www.wilytech.com

BEA WebLogic DEVELOPER'S JOURNAL

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

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JASON WESTRA

EDITORIAL DIRECTOR
JEREMY GEELAN

MANAGING EDITOR
SEAN RHODY

EXECUTIVE EDITOR
GAIL SCHULTZ

EDITOR
NANCY VALENTINE

ASSOCIATE EDITORS
JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR
JENNIFER STILLEY

WRITERS IN THIS ISSUE
KARL BANKE, STEVE BUZZARD, LEWIS CIRNE, SCOTT DIETZEN, PETER HOLDITCH, APURB KUMAR, DOC MILLS, SEAN RHODY, KATHIRAVAN SENGODAN, DAVE MARTIN

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

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

PRESIDENT, SYS-CON EVENTS
GRISHA DAVIDA

CONFERENCE MANAGER
MICHAEL LYNCH

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

SENIOR CUSTOMER CARE/CIRCULATION SPECIALIST
NIKI PANAGOPOULOS

CUSTOMER SERVICE REPRESENTATIVE
SHELIA DICKERSON

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

Flying South

BY SEAN RHODY
MANAGING EDITOR

My neighborhood is home to a host of birds, many of which fly south during the winter months. With spring in bloom, I always look forward to the return of the various avian travelers who dart and weave all over the open fields near my home. That's the kind of migration I look forward to.

In the software world, there's a less appealing form of migration, one we can never truly get away from. That's the migration of code and servers from one version to the next.

Some migration efforts have been a direct result of the evolution of the EJB and J2EE specifications. In the earliest incarnations, much of the deployment mechanism was left up to the container vendors. While EJBs were, in theory, portable, for all practical purposes, it was an adventure to move them from one container to another – which we often had to do as organizations explored the emerging container vendors, back before it became a two-horse race.

XML deployment descriptors were a big step in the right direction, allowing the IDE vendors to do a better job of deployment to multiple platforms. Part of that was just standardization, but part of it was driven by other necessities.

Probably the biggest driving force in this evolution was the development and standardization of Container Managed Persistence. As containers became more sophisticated, and Container Managed Relationships and EJBQL evolved, the deployment descriptors and mechanisms changed accordingly. Fortunately, the J2EE specification is now a bit more stable, and updates come at a more benign pace. Unfortunately, that doesn't insulate us from the other source of migration – vendor version updates.

The J2EE application server platform has grown by a process of accretion – last year's differentiators are this year's core product. JMS is a great example of that. All vendors try to innovate, and introduce new features that are not quite part of any specification. JCA is a perfect example. BEA supported it from the outset, but added functionality (asynchronicity, among other things) and then lobbied the JCP to include it in the next release of the specification.

All of which is good, up to a point – the nightmare of migrating an application from one version of the software to another. In some cases, such as when major specification changes occur, it behooves an organization to examine its application and see if it's time for refactoring.

But in other cases, the move to the next version of software is more painful than it should be. For example, it seems the console changes each time we get a new release of WLS. That adds to administration headaches, even if you just want to redeploy the old application.

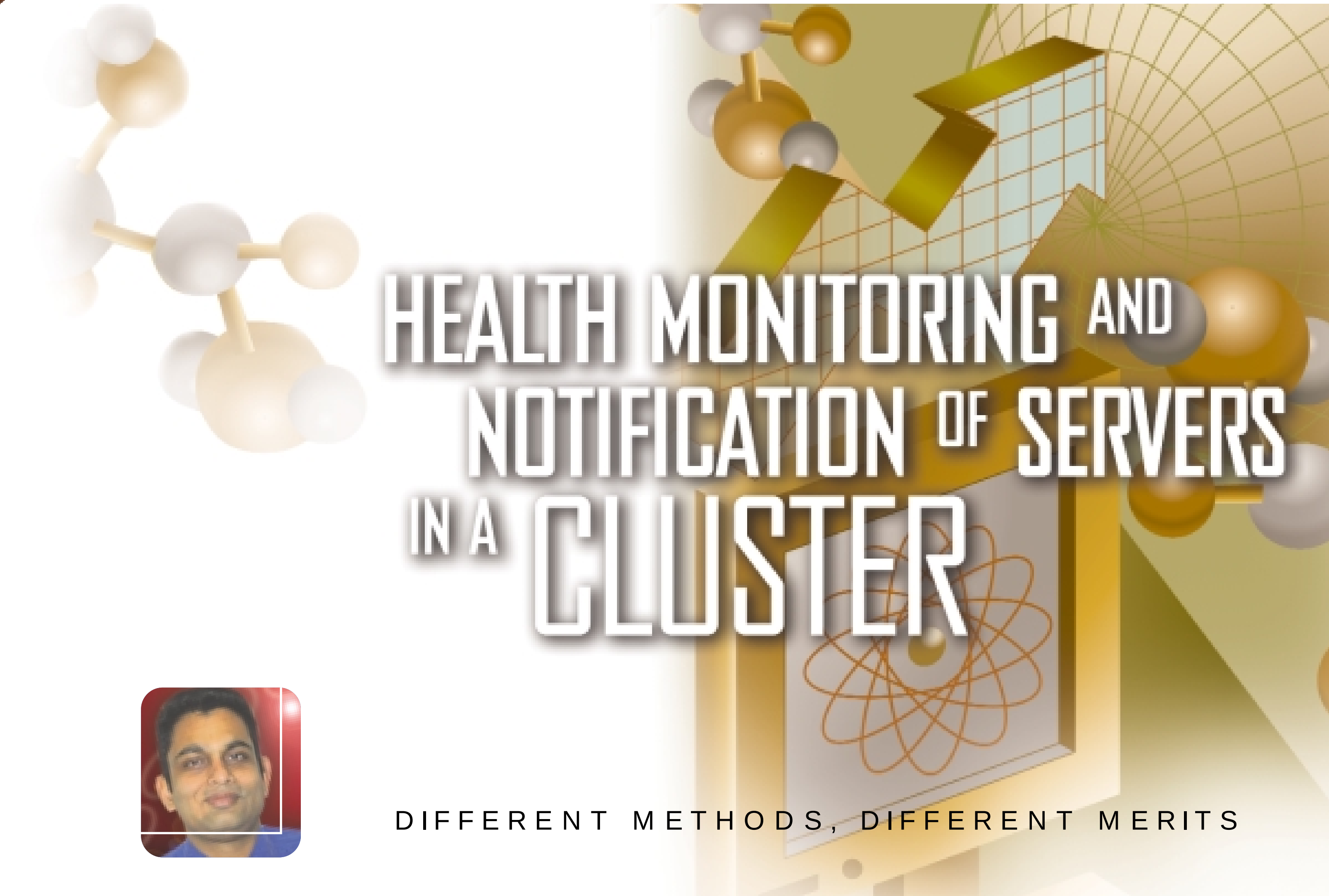
What we really need is to fix the deployment standard – with a common interface for the core specification. There should be a common descriptor, a common interface (or interfaces, one thick console, and swing based, one in JSP) that can be expanded, but not changed. Any new, proprietary functionality would be added as extensions, and managed by the console.

In the meantime, we have a new release of WebLogic Server. The early version that I've seen has a great deal to offer, even if it will take work to migrate to it. In particular the revamped Workshop is going to give MS Visual Studio a run for its money.

Even the bird migrations are not without a downside. As I look out my window, I see the turkey buzzards are back too. Still, I look forward to seeing most of the migrating creatures. Migrations are like that, even the best ones.

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

CONTACT: sean@sys-con.com



HEALTH MONITORING AND NOTIFICATION OF SERVERS IN A CLUSTER

DIFFERENT METHODS, DIFFERENT MERITS

BY **APURB KUMAR**

AUTHOR BIO...

Apurb Kumar is a developer relations engineer in Backline WebLogic Support at BEA Systems. He has more than 10 years of industry experience, starting with real-time programming, moving on to databases, and finally Java development. Before moving to BEA Systems, Apurb consulted for companies such as Charles Schwab, AllAdvantage.com, and Holland Systems.

CONTACT...

apurb.kumar@bea.com

What would happen if you had a stand-alone server, say an Admin Server having just a managed server? Or what if the Admin server itself becomes unresponsive. How would someone be notified when they became unresponsive?

"…..This is bad. Why did the server hang? And on top of this, why did we come to know about this so late?" As Bob heard his boss say this, he knew what was about to come next. He would be told to open a case with WebLogic support, who would help them do a postmortem of why and how the server stopped responding in production. Probably he would need to go beyond it this time, and figure out a way to automatically check server status.

I've often come across situations in which customers want to monitor the health of the servers running in a cluster. Better still, to get notifications should a condition arise. If you're running in a cluster, which of course most production-based systems do, the failover would take place right away, as soon as a server becomes unresponsive. As far as your applications go, there would be other servers that would serve the requests. But as an administrator you should be aware of this immediately, and take corrective measures. Again, what would happen if you had a stand-alone server, say an Admin server having just a managed server? Or what if the Admin server itself becomes unresponsive. How would someone be notified when this happens?

The answers to these questions can be either external or internal (meaning that you don't use any WebLogic-specific tool) to WebLogic. The external solution is probably the simpler one, as it makes use of a WebLogic ping utility that would result in a success if it finds the server running, and return a java.net.ConnectException if it isn't. Other than that, you pretty much rely on the OS script (and the network of course!) to do the job for you. The internal solution delves deeper into the WebLogic core, such as using SNMP and utilizing MBeans, specifically the ClusterMBeans.

Let's dive into the simple solution right away.

Our test case is: we are having running instances (they need not be in a cluster) of *x* servers. We want to be notified if any of them becomes unresponsive or goes down. All that you might be required to do is to execute the following simple steps:

- Set the environment so that the WebLogic-specific classes are in the classpath.
- Input the host:ports for all of the instances of WebLogic server that are running to the weblogic.Admin ping utility.
- Loop infinitely, pinging each instance (hosts:port) of the server passed as input.
- Define a time interval for the ping in the script so that after pinging each instance passed to weblogic.Admin, you sleep for some time.
- E-mail when a ConnectException is detected.

A simplified version of this utility running on Bourne shell on Solaris 5.6 is given in Listing 1 (the code for this article is available online at www.sys-con.com/weblogic/sourcec.cfm).

So all you might need to do is run the script in Listing 1 in the background, and it would keep you notified by e-mail should any server go down or become unresponsive.

Before we proceed with the MBean material a brief outline of how WebLogic Server instances in a cluster detect failures of their peer server instances would be helpful. Some of the information given here is from the WebLogic documentation. For details refer to the cluster specific information at http://edocs.bea.com/wls/docs61/cluster/index.html.

The instances monitor the:

- Socket connections to a peer server
- Regular server "heartbeat" messages

Failure Detection Using IP Sockets

WebLogic Servers monitor the use of IP sockets between peer server instances as an immediate method of detecting failures. If a server connects to one of its peers in a cluster and begins transmitting data over a socket, an unexpected closure of that socket causes the peer server to be marked as "failed," and its associated services are removed from the JNDI naming tree.

The WebLogic Server "Heartbeat"

If clustered server instances don't have opened sockets for peer-to-peer communication, failed servers may also be detected via the WebLogic Server "heartbeat." All server instances in a cluster use multicast to broadcast regular server "heartbeat" messages to other members of the cluster. Each server heartbeat contains data that uniquely identifies the server that sends the message. Servers broadcast their heartbeat messages at regular intervals of 10 seconds. In turn, each server in a cluster monitors the multicast address to ensure that all peer servers' heartbeat messages are being sent.

If a server monitoring the multicast address misses three heartbeats from a peer server (i.e., if it doesn't receive a heartbeat from the server for 30 seconds or longer), the monitoring server marks the peer server as "failed." It then updates its local JNDI tree, if necessary, to retract the services that were hosted on the failed server.

In this way, servers can detect failures even if they have no sockets open for peer-to-peer communication. In our case, the AliveServerCount value for each server would be the updated list of those numbers of active servers, which are still in the cluster JNDI list of the servers.

The next solution (if you don't want to use the shell script and for non-Unix platforms) is to use the WebLogic APIs to generate the trap and the Java Mail API for generating the notification. In other words, do something like this:

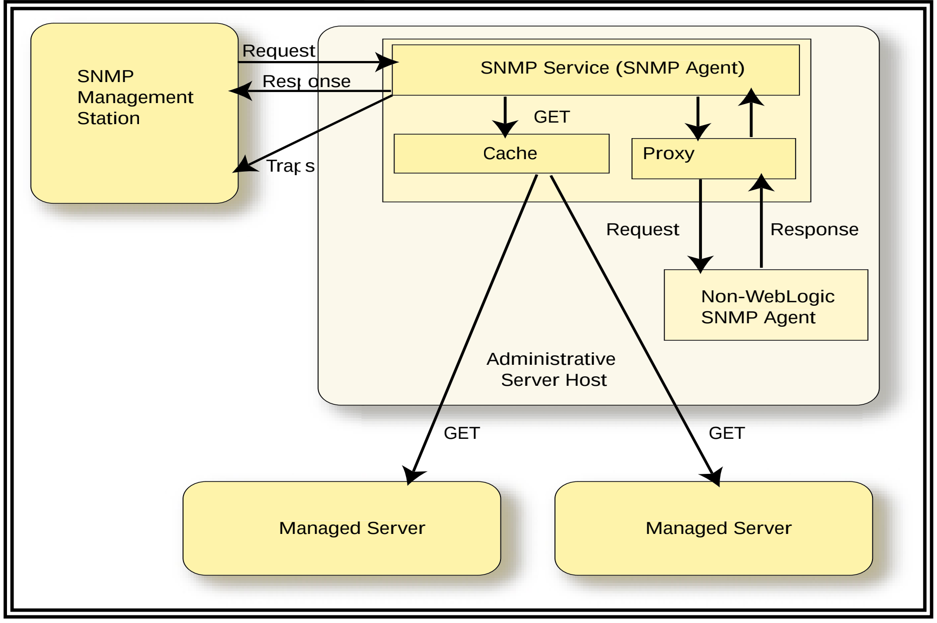
- Generate a list of arguments passing the host:port, username, password for the Admin server, the total number of servers participating in the cluster, and the delay interval acceptable for generating the e-mail. This list of arguments will be required by the Java program.
- Get the Admin MBeanHome by passing these specific properties.
- If the Admin home is not found, generate an appropriate error message.
- Use the Admin MBeanHome to get the ClusterRunTime MBean and iterate through to get the server names.
- Check to see if the total number of alive servers is less than the total number of servers passed as an argument to the Java program.
- If the count goes below the one passed as an argument, formulate the appropriate string that would be passed as the e-mail content.
- Generate an e-mail at the address specified for the SMTP (not to be confused with SNMP) Server.

Listing 2 would be the meat of this solution.

SNMP Model

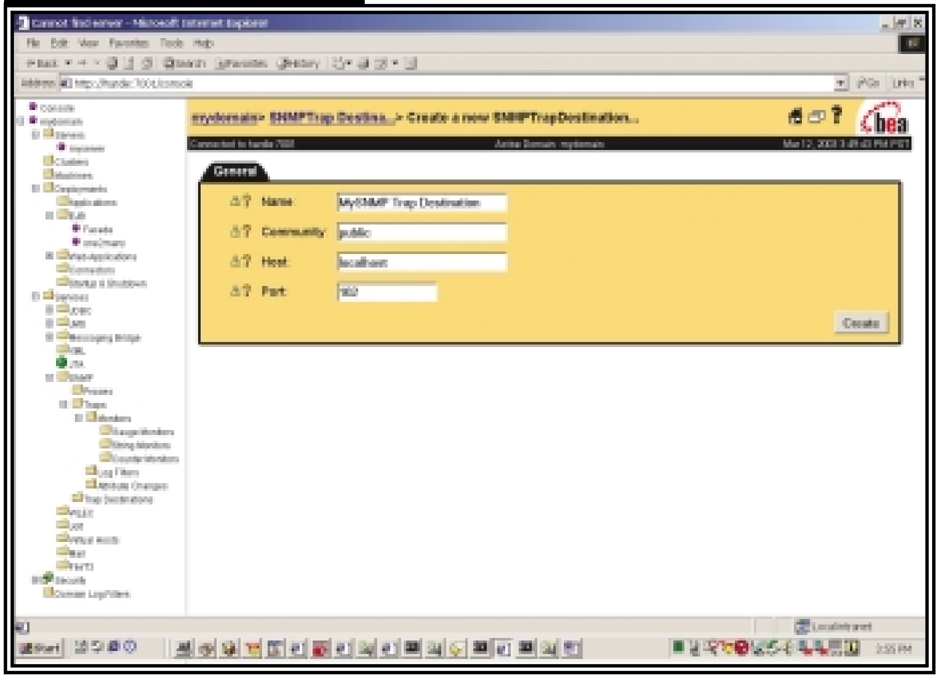
We now come to the last method that I'm proposing: using an SNMP model. WebLogic Server software includes the ability to communicate with enterprise-wide management systems using Simple Network Management Protocol (SNMP). The WebLogic Server SNMP capability enables you to integrate management of WebLogic Servers into an SNMP-compliant management system that gives you a single view of the various software and hardware resources of a complex, distributed system.

FIGURE 1



SNMP Management of a WebLogic Domain

FIGURE 2


mydomain> SNMPTrap Destina...> Create a new SNMPTrapDestination...

General

Name: MySNMP Trap Destination

Community: public

Host: localhost

Port: 162

Using the EBCC to define a Webflow

The following definitions help us derive a practical scenario for cluster monitoring and have been partially derived from the WebLogic documentation.

SNMP management is based on the agent/manager model described in the network management standards defined by the International Organization for Standardization (ISO). In this model, a network/systems manager exchanges monitoring and control information about system and network resources with distributed software processes called agents. In our case, the SNMP agent is the WebLogic Admin Server. For the SNMP manager as an illustration, and example, I used a freely downloadable third-party software called the “Trap Receiver”.

Any system or network resource that is manageable through the exchange of information is a *managed resource*. This could be a software resource such as a Java Database Connectivity (JDBC) connection pool or a hardware resource such as a router. In our case, we are monitoring the ClusterRuntime.

The underlying idea is that the WebLogic Admin server, which is acting as the SNMP agent, would act as a “collection device” that would gather and send us the information of the managed resource, i.e., the ClusterRuntime. This would be achieved by setting thresholds (referred to as Monitors) for any specific attribute for the ClusterRuntime. In our example we would monitor the AliveServerCount attribute for the ClusterRuntime. Say that the total servers running in a cluster is three; if any one of the servers becomes unresponsive, the AliveServerCount would decrease to two and a trap notification should be sent to the SNMP manager, which would then generate an e-mail.

The Trap Receiver relies upon a database of definitions and information about the properties of managed resources and the services the agents support – this makes up the Management Information Base (MIB). In our case, the MIB will be available under the WEBLOGIC_HOME/lib/ BEA-WEBLOGIC-MIB.asn1 (see Figure 1). (For more information about SNMP management, visit http://e-docs.bea.com/wls/docs61/snmpman/index.html.)

The following basic steps are required for setting up our Failure Notification Model:

- Configuring the SNMP Agent
- Configuring the SNMP Manager

Configuring the SNMP Agent

This would be the WebLogic Admin Server. We would start by assuming that we are running a cluster of three managed servers – managedserver1, managedserver2, managedserver3 – all of them listening at different IPs and the same port. The steps in this process will be:

- Access the 6.1 WebLogic Admin browser console after starting your Admin server.
- Click the Trap Destinations node on the left-hand pane.
- Fill in the appropriate values (see Figure 2)
- Click the Trag Destinations node on the left-hand pane after expanding the SNMP node.
- Click on the domain name on the left-hand pane and select the SNMP tab.
- Make sure that the Enabled check box is checked.
- Select the Trap Destination that you configured in the previous step as a target.
- The default value for MIB Data Refresh Interval is 120, and the least possible value is 30 secs. The MIB Data Refresh Interval is the interval, in seconds, at which the SNMP agent does a complete refresh of the cache. This value would eventually determine the freshness of data; for our case, the time since the last time the number of active servers was checked. Decreasing this value significantly might impact performance.
- If you want to use the default trap that Weblogic server generates when it goes down (OID 1.3.6.1.4.1.140.625.100.70, explained in detail in the SNMP Agent section), you need not follow any further steps for configuring the SNMP Agent and can jump directly to "Configuring the SNMP Manager."
- Expand the Monitors node on the left hand pane
- Click on “Configure a new gauge monitor" in the right-hand pane.
- Fill in the values shown in Figure 3.

The definitions for what the value stands for can be seen by clicking on the ‘?’ against each parameter. In our case, we would be creating three such monitors: MyGaugeMonitor1, MyGaugeMonitor2, and MyGaugeMonitor3. You would need to replace the name for the Monitored MBean Name with the names of your managed servers, respectively. The idea is that the SNMP agent will generate a trap whenever the value of the AliveServerCount for any server goes to two or below two. It will also generate a trap when the AliveServerCount goes to three or more, but in this specific case that information won’t be useful. You will need to apply the changes and select the Servers Tab to target the respective servers.

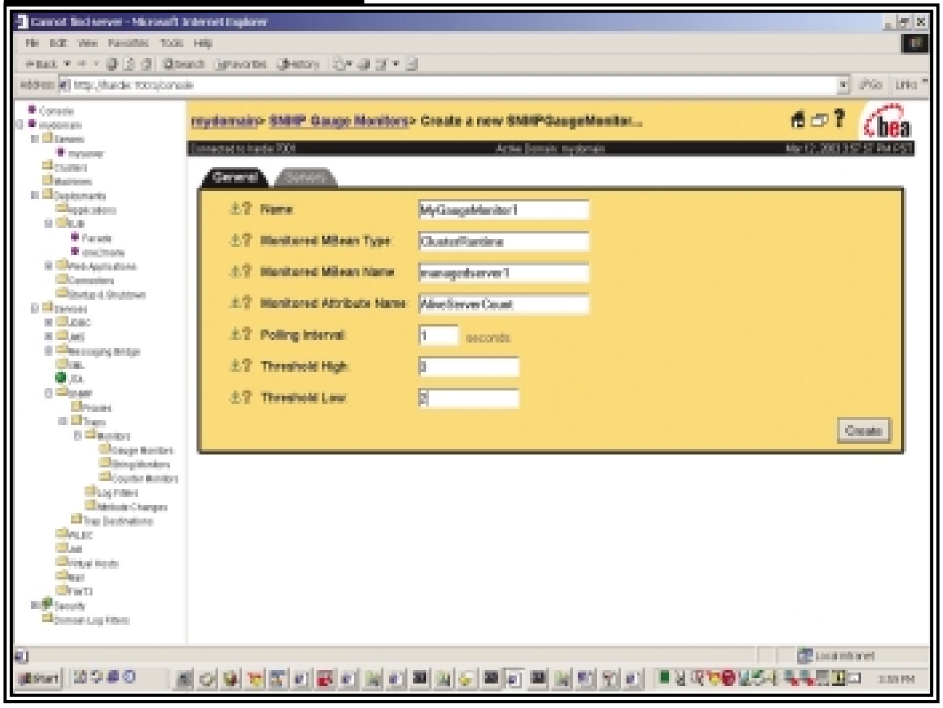
Configuring the SNMP Manager

As I mentioned earlier, I used a third-party tool, freely downloadable online from www.ncomtech.com/download.htm. I selected “Trap Receiver for NT/2000”, as I wanted this installed on my local Windows

PANACYA

www.panacya.com

FIGURE 3



mydomain> SNMP Gauge Monitors> Create a new SNMPGaugeMonitor...

Name	MyGaugeMonitor1
Monitored MBean Type	ClusterRuntime
Monitored MBean Name	managedserver1
Monitored Attribute Name	AliveServerCount
Polling Interval	1 seconds
Threshold High	3
Threshold Low	2

Creating a separate monitor

2000 box. Detailed help for the various attributes are available at www.ncomtech.com/trmanual.html. The following steps are required to quickly configure the SNMP manager after installation:

- Select the MIBs tab, hit Load, and select the location of WebLogic-specific MIBs, i.e., WEBLOGIC_HOME/lib/BEA-WEBLOGIC-MIB.asn1.
- Select the Actions tab, hit Add, and select the Varbind OID from the Watch drop-down.
- Fill in the actual MIB value for the trap in the Equals column. In our case it would be 1.3.6.1.4.1.140.625.100.75. Here 1.3.6.1.4.1.140.625 is the Enterprise vendor identification (OID) for the WLS6.1 instance we are using. The value of 75 is meant for the Monitor Trap we are interested in. We want to generate an e-mail when this value is reached so select the checkbox for the e-mail option.
- It is worthwhile to mention here that if we had wanted notifications to be generated using the default WebLogic traps, we don't need to create a separate monitor. There are some predefined WebLogic SNMP traps that would be generated automatically. For example, the server startup trap is 65, and the shutdown trap is 70. So the full Varbind OID would be 1.3.6.1.4.1.140.625.100.65 and 1.3.6.1.4.1.140.625.100.70, respectively. Hence if we don't want to go through the process of creating a separate monitor (in Figure 3 for MyGaugeMonitor1), all we need to do is fill the value of 1.3.6.1.4.1.140.625.100.70 instead of 1.3.6.1.4.1.140.625.100.75 in the previous step. I found that the trap notification was almost instantaneous in case of a simulated hang when using the custom created MyGaugeMonitor1, whereas it took a little while before the shutdown trap was generated. However, the disadvantage in this case would be that the number of e-mails that will be generated when the trap condition is reached would depend on the number of servers for which the monitor has been created minus the server, which went in hung state (in our case, 3 –1 = 2).
- We could have used the counter monitor here as well, in which only one of the thresholds would be required to be given. But again, this is only in case the value equals or exceeds the threshold value. The advantage of using the Gauge monitor was that the counter is reset after the stopped server, is restarted and the high value is reached.
- The last thing to be done is to configure the e-mail option. Select the e-mail tab and fill in the appropriate values for your SMTP server. For the message box, you might want to give something like "Startup/Shutdown/Hang. A trap from %SENDERIP% of type %GENERIC-TYPE%/%SPECIFICTYPE% was received".

Testing the SNMP Setup

We're now all set. When all three servers are up and running, the high value of the gauge is reached. This would generate an e-mail notification; however, this is not important to us. What we want is to be notified if the server becomes unresponsive or goes down. Bringing down one instance of managedserver would again generate a trap, and since the AliveServerCount would become 2, will generate the trap so that the e-mail notification can be sent. It you had created the noncustom monitor and used the default OID, you would still be notified. The purpose of using the custom monitor is to illustrate the usage of MBeans and monitors.

Summary

All of the options given here have different merits. The method using the WebLogic ping utility does not have any additional overhead as far as the performance of WebLogic server is concerned, whereas when the SNMP traps are used there could be a slight performance impact if the sampling period is reduced. On the other hand, in the first method, if you want to use the shell script as is (see Listing 1), it would benefit the Unix platforms. But, you would not get the real time information, and would have a separate share of CPU usage. The emailServersRunningInCluster.java (see Listing 2) can be set as a cron job on Unix platforms in case the sleep is eliminated in the code, and utilizes WebLogic-specific API,s. However, as it's pure Java you can run it from anywhere, provided the WebLogic-specific classes are present in the classpath. This would again have its separate share of CPU utilization. The third option, using the SNMP Manager, can be utilized for catching other traps as well, in addition to finding the AliveServerCount, by tweaking the OIDs and creating separate monitors. The Java program could not be used as is, and if you wanted to monitor other MBeans you would need major code changes.

A sequel to all this can be to find the Java process ID of the server that becomes unresponsive, either as a matter of ConnectException or AliveServerCount going down, and automate the script to do a kill –3 for Unix-based platforms to get the Thread Dumps and send it to support the analysis.

Borland

www.agile.borland.com/wd1

WEB 2.0: XML & JAVA-STANDARD INTEGRATION

BY **SCOTT DIETZEN**

AUTHOR BIO...

Dr. Scott Dietzen is chief technology officer of BEA Systems, Inc.

CONTACT...

scott.dietzen@bea.com

Standards can drive revolutionary changes in technology: consider the impact that SQL has had on the database market, or consider that the World Wide Web was launched by the combination of HTML, HTTP, URL, and SSL. Our belief is that protocol standards (XML, Web services) and programming standards (Java and the .NET alternatives, XML Query, etc.) will have a similarly profound impact on integration.

Integration encompasses a broad range of information technology (IT) needs (see Figure 1):

- ***Enterprise application integration (EAI):*** Directly interconnecting two or more business applications over intranets (that is, behind corporate firewalls)
- ***Business-to-business integration (B2BI):*** Directly interconnecting the business applications of one company with those of a business partner across the Internet or a virtual private network
- ***User interface (UI) integration:*** Aggregating and personalizing UI across applications so that the human user experiences a single unified "Web flow" (e.g., Java page flows within a portal), bringing together those disparate back-end systems
- ***Data integration:*** Aggregating data queried from disparate applications and data stores in order to define a unified view of a business object – for example, creating a unified "customer" schema by combining data from a customer relationship management (CRM) system, an Enterprise Resource Planning (ERP) system, and so on

Today, the integration market remains largely fragmented across proprietary solutions, none of which scale to meet the needs of our Web-oriented world. (*Note:* Consider that proprietary integration technologies generally require the user to run a particular vendor's software on all participating nodes. Convincing a worldwide company and all of its business partners to run the same proprietary integration software just will not fly.) At the same time, integration remains very expensive – by various measures consuming 70+% of discretionary IT funding, and remains sufficiently complex that organizations tend to only integrate applications as it becomes a business imperative.

Clearly, the industry must do better. We are convinced the answer lies in extending the Web from the UI platform it is now (browser to data/application) to become an integration platform. Today, most new applications are "Web ready" out of the box – i.e., designed to support a Web browser front end. Our claim is that future applications will also be "Web integration ready" out of the box – that is, ready to plug into this emerging "backplane" of eXtensible Markup

BETTING ON A PARADIGM SHIFT

Language (XML) and Web services that will enable them to seamlessly interconnect with other Web-based applications.

But XML and Web services only define the protocols – that is, the standard wire formats that ensure interoperability. How will the industry protect investment in the programming of the orchestration logic that ties Web applications together? Just as application programming standards like servlets, JavaServer Pages (JSP), and Active Server Pages (ASP) were crucial to the success of the Web, *integration programming* standards are essential to Web-based integration.

In fact, most of the infrastructure standards necessary to enable such a "sea change" are already in place. As a result, the integration market, like that for Web application development before it, is poised for sweeping standardization. BEA has been collaborating with our platform competitors (primarily IBM and Microsoft) as well as our allies to remake the Web into an application development *and* integration platform – Web 2.0 if you will! Assuming we are successful (as industry analysts now predict), Web application servers (for Web application development and hosting) will be superceded by *Web application platforms*, which include tightly integrated portal, EAI, B2BI, and data integration technologies. Web 2.0, then, may well have a bigger impact on enterprise IT than Web 1.0 did!

Integration Technology Stack

The stack of technologies required for integration is relatively well understood. Our proposed factoring appears in Figure 2. Let's consider each of the layers in turn (bottom up).

Middleware Infrastructure

The foundation of all integration products (including the more traditional proprietary technologies as well as the emerging Web platform alternatives) is the middleware bus that provides interconnections and ensures quality of service. Historically, the industry has selected industry-standard middleware (either Java 2 Enterprise Edition/[J2EE] or .NET-based) for developing and hosting Web applications, but proprietary middleware for integrating those applications. But the promise of Web 2.0 is based on convergence to a single unified platform that does both. As a result, the standard Web application servers (WebLogic, WebSphere, .NET, etc.) are displacing proprietary integration middleware: it turns out to be a lot easier to extend standard Web application servers to do integration than it is to remake proprietary integration brokers into Web-standard application servers. (*Note:* The fact that most of the historically proprietary integration vendors are now including application server technology within their next-generation integration solutions is more compelling evidence for this convergence. The challenge for those vendors, of course, is that this is the "home court" that the Web application platform leaders [BEA, IBM, Microsoft] staked out years ago.)

Extra-Application Connectivity

Web services and adapters link the Web integration platform to other applications and data sources. While the middleware layer is colored green (meaning that J2EE and .NET are now safe bets), this layer is a mix of red and green since the essential Web services standards are still being fleshed out.

XML/WEB SERVICES

Web services are simply conventions for passing XML between applications. Figure 3 illustrates the *when* and *where* of Web services – Web services should be preferred for *inter*-application interoperability. The art is in defining the ideal *coarse-grained*, long-lived business interfaces that best encapsulate the application. Business application architects should carefully review such candidate Web services.

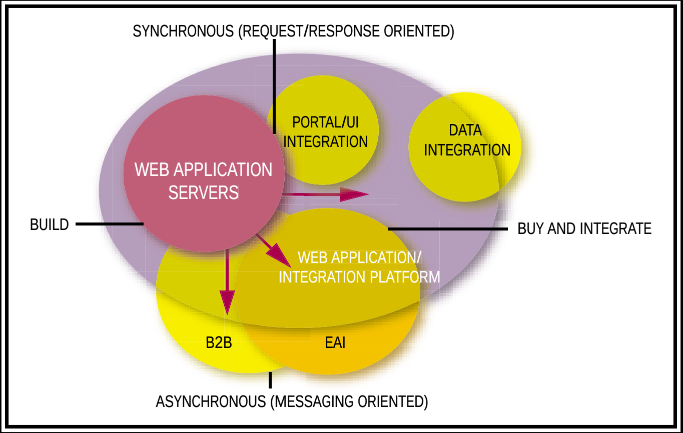
Our take on the emerging suite of XML/Web services standards (a.k.a., acronym soup) is depicted in Figure 4. As with the Web stack before it, we propose an essential core set of standards from which the "integration Web" will reach critical mass.

We include within this core the technologies that Web Services Interoperability (WS-I) validates in its basic profile. (*Note:* Today *inter*-container interoperability for Web services should likely be restricted to the WS-I *basic profile* of XML Schema, SOAP 1.1, and WSDL 1.1. By venturing outside the scope of what we vendors are ourselves testing, you are likely to encounter interoperability issues between different containers – e.g., between WebLogic and .NET. *Intra*-container interoperability, however, should not be an issue for the newer Web service standards.)

An XML-based schema language, typically XML Schema, is the preferred choice for Web service payloads. If XML Schema is becoming a lingua franca for interoperable business objects, then the Simple Object Access Protocol (SOAP) is the framework of "composable" message headers for defining ever higher qualities of service for passing those business objects between applications (e.g., security, guaranteed delivery).

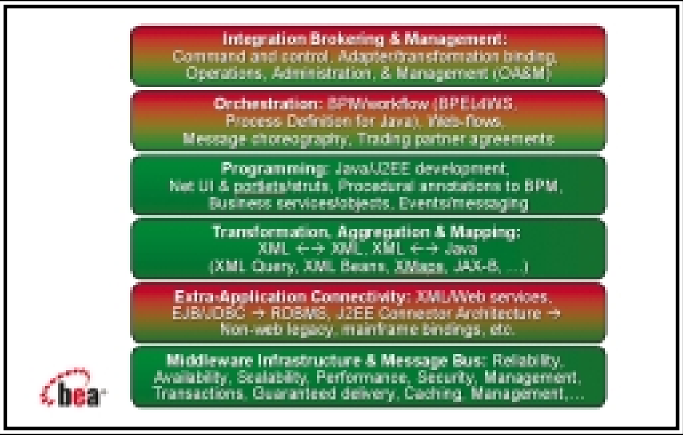
With Web services, programmers should be careful not to "hardwire" schemas to the business logic so as to achieve the looser coupling that made the Web so successful. Unfortunately, loose coupling is not inherent in Web services – you must program for

FIGURE 1


SYNCHRONOUS (REQUEST/RESPONSE ORIENTED)
PORTAL/UI INTEGRATION
DATA INTEGRATION
WEB APPLICATION SERVERS
BUILD
BUY AND INTEGRATE
WEB APPLICATION/ INTEGRATION PLATFORM
B2B
EAI
ASYNCHRONOUS (MESSAGING ORIENTED)

Web application platform evolution

FIGURE 2


Integration Brokering & Management: Command and control, Adapter/transformation binding, Operations, Administration, & Management (OA&M)
Orchestration: BPM/workflow (BPEL4WS, Process Definition for Java), Web-flows, Message choreography, Trading partner agreements
Programming: Java/J2EE development, Net UI & portlets/struts, Procedural annotations to BPM, Business services/objects, Events/messaging
Transformation, Aggregation & Mapping: XML ←→ XML, XML ←→ Java (XML Query, XML Beans, XMaps, JAX-B, ...)
Extra-Application Connectivity: XML/Web services, EJB/JDBC → RDBMS, J2EE Connector Architecture → Non-web legacy, mainframe bindings, etc.
Middleware Infrastructure & Message Bus: Reliability, Availability, Scalability, Performance, Security, Management, Transactions, Guaranteed delivery, Caching, Management,...
bea

Application integration stack

it. (*Note:* Loose coupling works on the Web in the sense that a Web site can undergo dramatic change [say a migration from .NET to Java] without affecting the end users [who may not even notice ASPs turning into JSPs]. Web services loose-coupling is fundamentally harder to achieve. The Web client [browser] and schema [HTML] are fixed, while both sides of the Web services "wire" are expected to accommodate change independently.)

In particular, we recommend that if you use Java as your Web services "design center" (that is, you autogenerate Web Services Definition Language [WSDL] from Java classes), you should use "wrapper" classes rather than directly exporting application objects. We also recommend that you leverage the extensibility model of XML Schema, and use higher-level bindings like XML Query and "schema compilers" such as XML Beans, JAX-B, and JAX-RPC. Developers can thereby ensure that application changes will be less likely to break Web service interfaces and interface changes will be less likely to break applications.

Above the basic profile defined by WS-I are our proposals for the standards that will complete the Web services core. WS-Security provides selective encryption (privacy), nonrepudiation, and support for delegation (propagation of security context). This emerging OASIS standard is poised to be the "SSL" of Web services.

WS-Reliable Messaging (WS-RM) and WS-Addressing (from BEA, IBM, Microsoft, and others) ensure that message delivery can be *transactionally guaranteed* across unreliable networks and systems. WS-RM and WS-Addressing use *store and forward messaging* to commoditize into the Web services stack what has historically been the domain of proprietary message-oriented middleware like MQSeries. BEA remains convinced that such asynchronous Web services are the "sweet spot" for transactions/updates, while synchronous Web services tend to be better suited to queries. This is why we have put so much energy into making asynchronous Web services as easy to program as synchronous ones.

ADAPTERS

Adapters solve the "last mile" problem of integration: they provide the link from the Web technology to a non-Web or "legacy" technology (such as COBOL/CICS). (*Note:* By using the term "legacy," I intend no insult: becoming part of the legacy is the highest goal that software can achieve [of course, many of them never make it], and in a very real sense, all production applications are legacy.) Even with the emergence of XML and Web services, adapters remain essential because so little of today's legacy applications will be directly extended to support Web services. (Instead, legacy applications will be "wrapped" with Web services and new business logic deployed on containers like WebLogic, WebSphere, and .NET.)

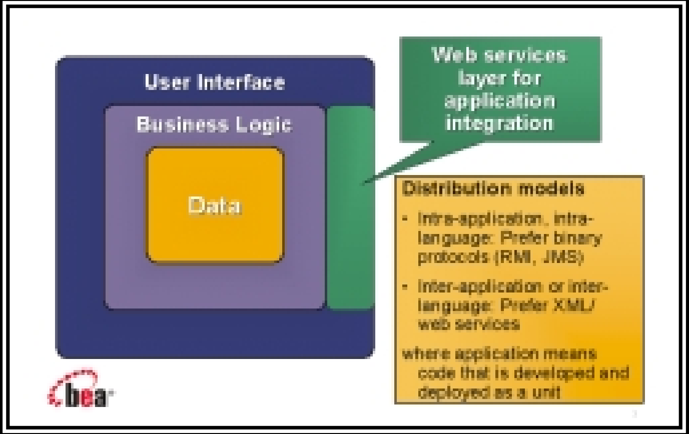
Without a standard model for adapters, it's nearly impossible for an integration platform to get to critical mass. Consider the impact that standardizing a common database adapter – Java Database Connection (JDBC) – had on Java's success. The J2EE Connector Architecture (CA) generalizes JDBC to define a universal model for Java adapters. J2EE CA (like the .NET alternative) eliminates the *m x n* cost of each integration vendor (*m*) having to "one off" their own proprietary adapter for each and every legacy system (*n*). Moreover, J2EE CA protects your programming investment in integration solutions just as JDBC protects investment in database applications.

Transformation, Aggregation, and Mapping

Inherent in our notion of Web 2.0 is the expectation that XML becomes the inter-application data representation. Despite the marketing hype to the contrary, XML does not itself resolve impedance mismatches between differing data representations, and we are witnessing the proliferation of XML Schemas across the enterprise. (Some large companies may already have more than a hundred schemas for "customer".) XML simply defines a common alphabet for constructing documents (think "structured ASCII"); it is still up to the respective companies and industries to define the vocabularies (the semantic mapping from XML to business objects) that enable conversations to take place. Some of these vocabularies will be defined "top down" by industry standards bodies and consortia. Other vocabularies will grow "bottom up" (much as natural languages have) as individual companies or small groups of companies specify the Web services necessary for conducting business with them. XML-based EAI faces the same challenges, albeit within a single company.

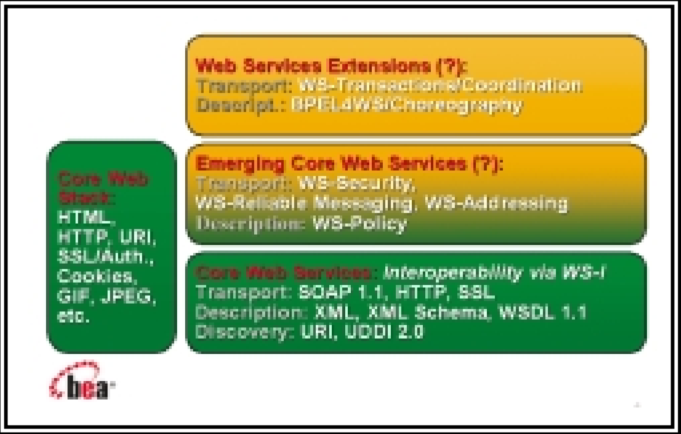
Tools that map from one vocabulary to another dramatically ease this burden – XML Query in particular is becoming the "SQL" of XML processing. But while XML transformation is getting easier and faster, you should nevertheless seek to constrain the explosion of XML Schemas by introducing a review/approval process by your business application architects.

FIGURE 3


User Interface
Business Logic
Data
Web services layer for application integration
Distribution models
- Intra-application, intra-language: Prefer binary protocols (RMI, JMS)
- Inter-application or inter-language: Prefer XML/ web services
where application means code that is developed and deployed as a unit
bea

Web services for service-oriented application architecture

FIGURE 4


Web Services Extensions (?):
Transport: WS-Transactions/Coordination
Descript.: BPEL4WS/Choreography
Emerging Core Web Services (?):
Transport: WS-Security, WS-Reliable Messaging, WS-Addressing
Description: WS-Policy
Core Web Stack: HTML, HTTP, URI, SSL/Auth., Cookies, GIF, JPEG, etc.
Core Web Services: Interoperability via WS-I
Transport: SOAP 1.1, HTTP, SSL
Description: XML, XML Schema, WSDL 1.1
Discovery: URI, UDDI 2.0
bea

Emerging Web services stack

BMC Software

www.bmc.com

Programming

The cost of integration solutions is still dominated by the cost of programming the glue code that ties the integration business process (workflows and Web flows) to the underlying business applications. Unfortunately, the state of the art still remains a long way from nirvana – when a business analyst will define only the high-level business process, and the infrastructure will "magically" generate the necessary glue code. Historically, one of the biggest risks to integration projects has been their dependence on proprietary application programming interfaces (APIs) for programming this glue. With no standard APIs, there is no investment protection in integration programs or programmers. Nor are there the richer tools that follow from API standardization (such as those created for SQL and J2EE). Independent software vendors (ISVs) in particular are desperate to build integration solutions that can be deployed across multiple containers (e.g., be portable to both WebLogic and WebSphere).

Of course, the Java and XML communities have been hard at work extending the Web application platform standards to protect investment in integration tasks. Some highlights worth a closer look:

- XML Query (W3C)
- Java Web Services (JWS) (JSR-181)
- Process Definition for Java (JSR-208)
- BPEL4WS (published by BEA, IBM, and Microsoft)
- Java Meta Data (JSR-175)
- XML Beans (open source from BEA)
- Portlets (JSR-168)
- Content Management Interface (JSR-170)
- Apache Struts and Java Server Faces (JSR-127)
- Web Services for Remote Portlets (WSRP; OASIS)

Orchestration

At the orchestration level, we include the declarative glue essential to integration: workflows, "Webflows," "work-lists" (collaborative document flow through an organization), and choreography – abstracted message sequences that define how workflows can be composed within and even between companies.

The most well understood technology at the orchestration level is Business Process Management (BPM). BPM refers to the graphical languages and tools for composing workflows – computational sequences of tasks and exceptions. We are convinced that a variety of workflow representation languages will be required by the enterprise:

- Those targeted at the business analyst (e.g., Aris)
- Those that are seamlessly integrated with Java (Process Definition for Java/JPD)
- Those that are programming language independent (BPEL4WS)

For Java-based workflows, PDJ will remain the best fit. BPEL4WS, however, provides a level of independence between the Java and .NET worlds, at the cost of introducing a new XML programming language. You should expect PDJ and BPEL to converge going forward – PDJ will become the Java realization of a BPEL workflow

Integration Brokering and Management

While integration solutions still require coding, the goal remains to minimize that coding. Security is similar in this regard: historically, authorization was hand-coded into the application. Now the leading Web platforms define security policies administratively through business-oriented rules (that can be applied to a particular endpoint or across an entire application).

The same shift is already underway for command and control – the decision about how to route a particular request (say a trade) could depend upon the client (quality of service obligation), the content (dollar value), the receiver (which transform or adapter to apply), or the state of the infrastructure (availability, load). If such logic is programmed into the application, then it cannot be changed without modifying and redeploying the business logic. By capturing such *metadata* administratively, change can be accommodated far more easily and economically.

The integration platform's management environment must provide a comprehensive operational view into the interworkings of the integration broker and its connections to various applications. Looking forward, integration environments will have to become more self-optimizing and self-healing – raising alerts to the human administrator only when the available resources are insufficient to meet the committed quality of service.

Conclusion

Addressing the complexity and cost of integration may well be IT's single biggest need. Unfortunately, there are no panaceas: integration is and will remain an inherently hard problem. However, with Web 2.0, the industry can still do dramatically better – better by increasing investment protection and by cutting noninherent complexity. With a standard integration platform and the associated tools, the industry should be able to cut noninherent complexity by a factor of 10 – making integration projects more predicable, more successful, and maybe even fun.

Of course, Web 2.0 is going to take time to mature. Web services standards, in particular, continue to evolve: interoperable end-to-end security and guaranteed delivery should appear later this year.

So if the Web 2.0 standards are not all in place just yet, does that mean organizations should wait? Hardly. Products that represent the leading edge in Web- and Java-standard integration – like WebLogic Integration, Portal, Liquid Data, and their direct competitors – are feature/function competitive with the proprietary alternatives today, and in a dramatically better position to protect your long-term investment.

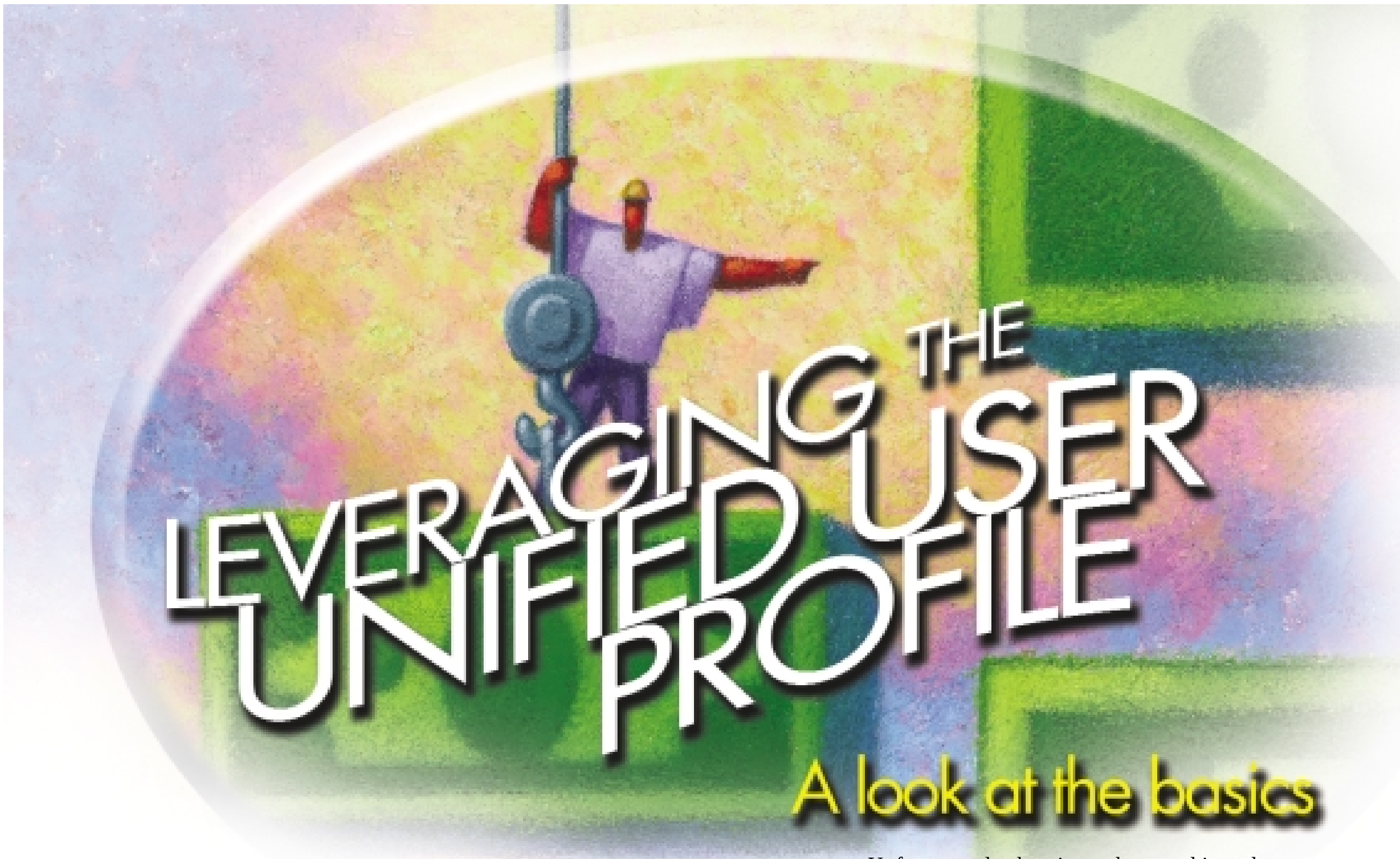
How, then, should organizations best prepare themselves for Web 2.0-based integration? By treating integration challenges both tactically and strategically: tactically, by getting the job done with a mix of best-fit standard and proprietary technologies; and strategically, by betting on the emerging Web 2.0 integration platform.

For those who still have doubts about integration converging around these standards, a little history: in 1997 we made a little-noticed prediction that an emerging mix of server-side Java standards (which later became J2EE) would have a profound impact on Web application development. At the time, there were some 30-odd Web application server vendors, each promoting their own proprietary programming models and tools. Today, there is a multi-billion dollar market for J2EE-standard Web application servers led by BEA WebLogic and IBM WebSphere, and Microsoft has launched the .NET Server family, at least partially in response to the popularity of J2EE.

The smart money is on history repeating itself for Web-based integration. The compelling need for a more scalable approach to integration, for investment protection via standards, and for the ease-of-development tools that follow from standards will drive this convergence. As with all technology transformations, right now there are compelling opportunities to gain a competitive advantage, especially for independent software vendors and systems integrators focused on integration. Of course, there is also risk. Yes, risk in betting on technologies that are not fully baked, but also risk in throwing good money after bad by not recognizing an impending paradigm shift.

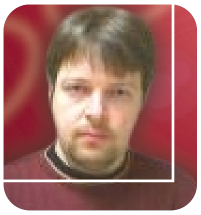
Wysdom

www.wysdom.com



LEVERAGING THE UNIFIED USER PROFILE

A look at the basics



BY **KARL BANKE**

AUTHOR BIO...

Karl F. Banke is principal consultant and general manager of iternum GmbH, Frankfurt, Germany. His working areas include J2EE and EJB architectures, Web Services and project management.

CONTACT...

karl.banke@iternum.com

In this article, I'll show you how to use the Unified User Profile in BEA WebLogic Portal. I look at the programming and configuration steps necessary to use the portal's built-in rules engine together with the external user profile data to provide a personalized portal presentation.

BEA WebLogic Portal runs on top of WebLogic Server and has a number of features and services that enable you to build an interactive and adaptive portal site. One of its great strengths is that it provides tools to configure the site based on the user's profile. For example, you can grant access to certain portal pages or even portlets only if the user's profile meets certain requirements. Or you can tailor the content presented within a portal page or a portlet based on the user profile. For example, a user may need to be a member of a certain division or to hold a certain position in the company to be able to access different content items or portlets. This is accomplished using rules that are interactively defined in BEA's E-Business Control Center (EBCC). Using this tool, it's easy to target content or marketing campaigns to certain users, without in-depth programming knowledge. On the other hand, as a software developer you enjoy the benefit of being able to access and modify the user profile using a convenient Java API and Taglib.

Unfortunately, data is rarely stored in only one place in the enterprise, nor is it usually in the precise format needed by BEA WebLogic Portal. In fact, profile data can be distributed between a lot of different systems. Often part of the data is stored in ERP systems like SAP, CRM systems like Siebel, LDAP Directory Services, different relational databases like Oracle or Sybase, and various other places. Luckily, BEA Portal doesn't only provide user profile storage in its native database schema. It defines a mechanism called the Unified User Profile (UUP) that enables you to create software modules that snap into the overall user profile of the portal application. You can then create content selection rules or user entitlements leveraging data that is already available and safely stored in your enterprise legacy or core systems.

Building a Unified User Profile

In this article I'll show you how to build and use a UUP that relies on external data. Let's consider an employee portal for a large multinational corporation. For simplicity, let's consider one portal page that contains a portlet showing corporate news and a portlet showing sales figures for the company's international divisions. The latter is available only to users working in the sales division. The news portlet presents corporate news based on the division an employee works in, their country, and thier function in the company. Some news will only be relevant and thus only be shown to software developers while others are only available to management. Of course, any information is presented in the preferred language of the employee. The corporation already stores most data about their employees in various data sources. Fortunately, this particular corporation has committed to a service-oriented software architecture, so all the relevant employee data is accessible as a SOAP Web service as defined in Listing 1 (Listings 1–3 are online at www.sys-con.com/weblogic/sourcec.cfm).

To use the data from the Web service in the portal, you must first define the user profile and the relevant rules using the EBCC. User profiles are created in the site infrastructure tab. As a general rule of thumb, for each data source that you access you should create at least one user profile. For the requirements, you'll want to define two user profiles. The first one, corporateProfile, is shown in Figure 1. This profile will later be mapped to the data obtained from the Web service. If you examine the WSDL for the Web service, note that it does not define the preferred language of the employee. Therefore, you need to create another profile called preferencesData that will be stored in the default portal database schema.

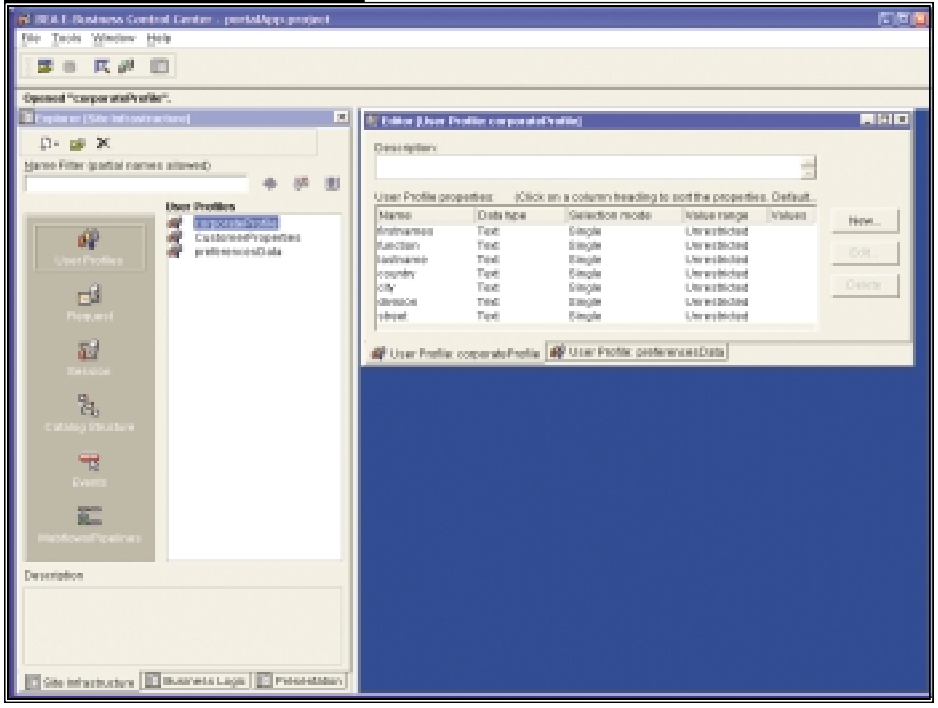
Next, you need to set up the rules that will be applied to the user profile later on. First, define the rule that selects the appropriate news content for the user. You need to create a new content selector in the presentation tab of the EBCC. Now, define the content selection rule. When defining these rules, the EBCC connects to your portal application to obtain the available content types and their metadata. The metadata that is actually available of course depends on the content loaded into your portal schema. For demonstration purposes, BEA provides a tool called bulk loader. This tool can be used to index and load content from specially annotated HTML files. Once you retrieve the metadata, you can build a content search using the available menus.

Figure 2 shows the rule that makes up the content search. It selects only the content items of type news that match the country, division, function, and language preferences of the user. Save the rule as "ShowNews" to access it later in the news portlet. In a similar fashion, you can create an entitlement segment that contains all users who work in the sales department and call it MemberOfSales. Upon synchronization, the rules are available for usage in the portal application.

Develop the UUP

Now for the actual software development work. For the rules to have the expected result, you need to provide the component that connects the user profile data from the Web service to the portal application. The general structure of access to the user profile is shown in Figure 3. In BEA WebLogic Portal, all access to the user profile is performed using a stateless session bean UserManager. Based on a user's profile type, it delegates access to an appropriate ProfileManager EJB. The ProfileManager Bean in turn delegates access based on the requested user profile to a stateless session bean of type com.bea.p13n.property.EntityPropertyManager. This performs the actual access to the user profile (see Figure 3). The framework exposes two hooks where you can insert your own profile access. In rare cases you will need to write a custom ProfileManager for a specific profile type. This is an option

FIGURE 1

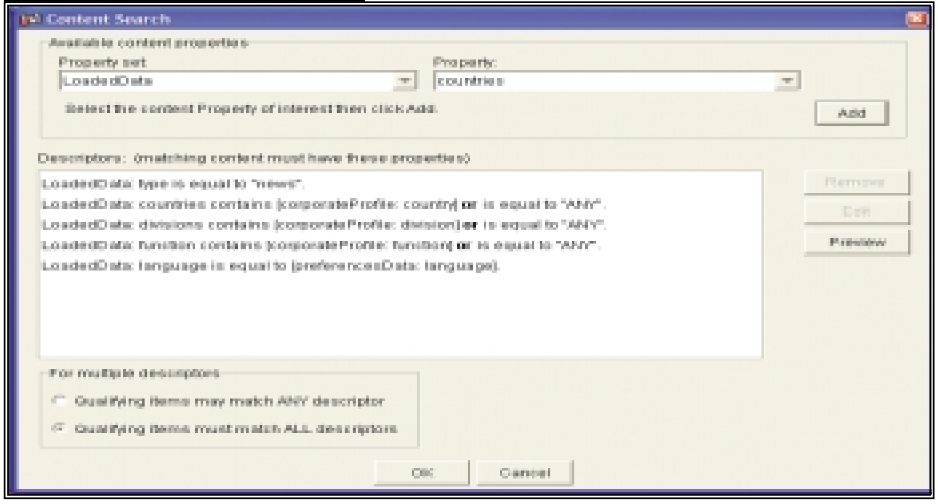


User Profile properties: (Click on a column heading to sort the properties. Default...

Name	Data type	Selection mode	Value range	Values
firstname	Text	Single	Unrestricted	
function	Text	Single	Unrestricted	
lastname	Text	Single	Unrestricted	
country	Text	Single	Unrestricted	
city	Text	Single	Unrestricted	
division	Text	Single	Unrestricted	
street	Text	Single	Unrestricted	

CorporateProfile

FIGURE 2



Property set: LoadedData
Property: countries
Select the content Property of interest then click Add.

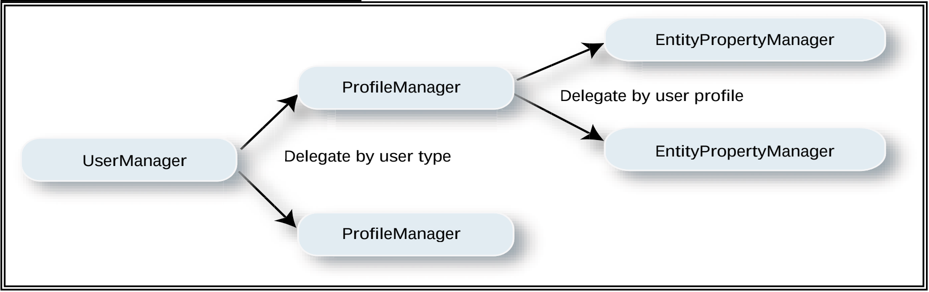
Descriptors: (matching content must have these properties)

LoadedData: type is equal to "news".
LoadedData: countries contains [corporateProfile: country] or is equal to "ANY".
LoadedData: divisions contains [corporateProfile: division] or is equal to "ANY".
LoadedData: function contains [corporateProfile: function] or is equal to "ANY".
LoadedData: language is equal to [preferencesData: language].

Qualifying items may match ANY descriptor [✓ Qualifying items must match ALL descriptors]

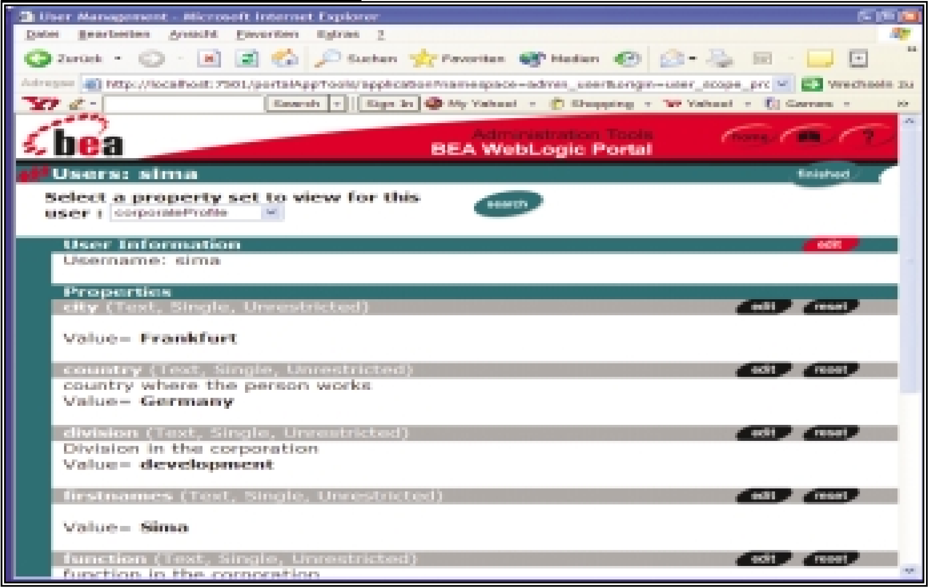
Content search

FIGURE 3



User management

FIGURE 4


User Management - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Adresse http://localhost:7501/portalAppTools/application?namespace=admin_user&origin=user_scope_pr
Wechseln zu
Search Sign In My Yahoo! Shopping Yahoo! Games
Administration Tools
BEA WebLogic Portal
home ? finished search edit reset

Users: sima

Select a property set to view for this user : corporateProfile

User Information
Username: sima

Properties
city (Text, Single, Unrestricted)
Value= Frankfurt

country (Text, Single, Unrestricted)
country where the person works
Value= Germany

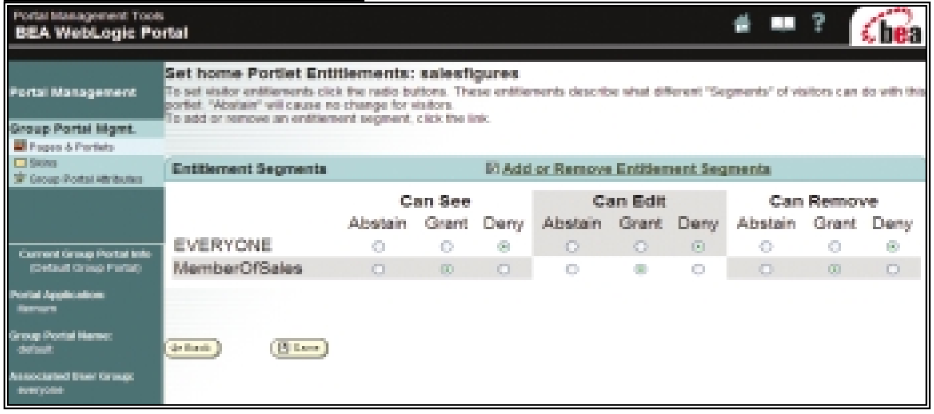
division (Text, Single, Unrestricted)
Division in the corporation
Value= development

firstnames (Text, Single, Unrestricted)
Value= Sima

function (Text, Single, Unrestricted)
function in the corporation

Profile check-in

FIGURE 5



Set home Portlet Entitlements: salesfigures

To set visitor entitlements click the radio buttons. These entitlements describe what different "Segments" of visitors can do with this portlet. "Abstain" will cause no change for visitors.
To add or remove an entitlement segment, click the link.

Entitlement Segments — Add or Remove Entitlement Segments

	Can See			Can Edit			Can Remove		
	Abstain	Grant	Deny	Abstain	Grant	Deny	Abstain	Grant	Deny
EVERYONE	○	○	●	○	○	●	○	○	●
MemberOfSales	○	●	○	○	●	○	○	●	○

Management entitlement segment on the sales portlet

if you need to provide an implementation that provides behavior that is fundamentally different from the default implementation. In most cases you'd want to provide a custom EntityPropertyManager and register it for a particular user profile defined in the EBCC. For the remainder of this article I'll focus on the latter scenario.

User Profile Definied in EBCC

You need to create an EJB where the remote interface is of type EntityPropertyManager, deploy it in the same EAR as BEA WebLogic Portal, and register it with the portal's user management. Listing 3 shows the classes that make up the CorporateProfileManager EJB. In our example, the user's data is read-only and cannot be changed using the portal's profile management. This is not an unusual condition in corporate environments, where such data may only be edited by authorized personnel. Under these circumstances, the only methods of EntityPropertyManager that must have a nonempty implementation are:

```
public Object getProperty(PropertyLocator
propertyLocator,
String pSet, String pName) throws
EntityNotFoundException;

public EntityPropertyCache
getProperties(PropertyLocator propertyLocator)
throws EntityNotFoundException;
```

For all other methods you should throw an UnsupportedOperation exception as shown in the code. Note that it is good practice to cache the results of retrieving the user profile for a sensible amount of time. Likewise, the getProperty() method usually delegates to the getProperties() method. This retrieves all user profile data for a particular user at once and puts it in the cache for subsequent use. The Web service that provides the data for the user profile has been created using BEA WebLogic Workshop. The CorporateProfileManager uses the stubs that have been created by WebLogic Workshop to access the service hiding any implementation details.

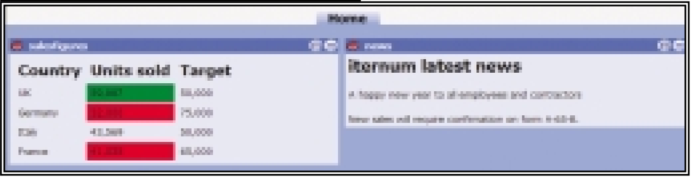
The EJB is packaged in its own JAR and deployed in the same enterprise application the portal resides in. Finally you need to tell the ProfileManager EJB that it should serve any request for user profile "corporateProfile" by delegating to the newly deployed CorporateProfileManager EJB. Mapping the user profile to the EJB in the ProfileManager's deployment descriptors does this. The ProfileManager is located in usermgt.jar. You can edit the deployment descriptor using the WebLogic console or by extracting the files from the JAR file, modifying them, and repacking the JAR. As shown below you need to add a new env-entry that maps the profile to the CorporateProfileManager. The name of this entry must match the pattern PropertyMapping/<profileName> , where <profileName> is the user profile name as defined in the EBCC. Also, you need to create an ejb-ref that references the CorporateProfileManager ejb you just created (see Listing 4).

Finally, you need to edit weblogic-ejb-

Quest Software

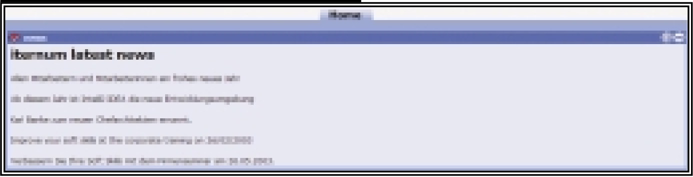
http://java.quest.com/jcsv/wldj

FIGURE 6a



Portal page for user Teresa, who works in Sales

FIGURE 6b



Portal page for user Sima, who works in Sales

jar.xml to map the ejb-ref-name created in web.xml to the actual JNDI name of the deployed EJB as shown below.

```
<weblogic-enterprise-bean>
    <ejb-name>UserProfileManager</ejb-name>
    <reference-descriptor>
…
      <ejb-reference-description>
        <ejb-ref-
name>ejb/CorporateProfileManager</ejb-ref-name>
        <jndiname>

${APPNAME}.BEA_personalization.CorporateProfileMa
nager
        </jndi-name>
      </ejb-reference-description>
    </reference-descriptor>
…
</weblogic-enterprise-bean>
```

After you apply these changes all you need to do is redeploy the enterprise application. Your user profile is now ready for use.

UUP in Action

Now let's see the new user profile in action. Consider two employees: Sima, who works as a manager in the development division in Frankfurt, Germany; and Teresa, who works as a clerk for the sales division in London, UK. Sima's preferred language is German, while Teresa's preferred language is English.

To check whether the profile is actually accessed in the way you intended, you can use WebLogic Portal's management WebApp. Select user management->users and select one of the users. Choose corporateProfile as the property set to view for the user. If everything works correctly, you should see the actual values that have been retrieved from the Web service (see Figure 4). Now create and deploy your portlets using the EBCC portlet creation wizards. Use the content selector you defined in the EBCC to select the appropriate news content to display to the users. A simple example of the news portlet is shown in Listing 3. The code that actually executes the content selector is in the following:

```
<pz:contentSelector rule="ShowNews"

contentHome="<%=ContentHelper.DEF_DOCUMENT_MANAG-
ER_HOME %>"
   id="news"/>
```

Note that the actual rule that we use accesses both the preferencesData and the corporateProfile to obtain the data. In effect, we can mix profile data from different sources to select content or to create user access rights. In the portalTools Web application, you need to make the portlets available and visible. Similarly, you need to apply the management entitlement segment on the sales portlet (see Figure 5).

Figures 6a and 6b show the portal display for users Teresa and Sima, respectively. As expected, Teresa sees the sales figure portlet since she works in the sales division. All her news items are in English according to her profile settings. Sima doesn't see the sales figures since she isn't a member of the sales division and is thus not entitled to see the portlet. Her news items are all in German. Also, she gets to see a number of different news items because she is in the development division.

Listing 4: ejb-ref that references the Corporate Profile Manager EJB<session>

```
      <ejb-name>UserProfileManager</ejb-name>
      ...
      <env-entry>
        <env-entry-name>PropertyMapping/corporateProfile</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>CorporateProfileManager</env-entry-value>
      </env-entry>
      ...
      <ejb-ref>
        <description>ejb/CorporateProfileManager</description>
        <ejb-ref-name>ejb/CorporateProfileManager</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>com.iternum.uup.CorporateProfileManagerHome</home>
        <remote>com.iternum.uup.CorporateProfileManager</remote>
        <ejb-link>CorporateProfileManager</ejb-link>
      </ejb-ref>
      ...
</session>
```

Summary

By now you have mastered and understood the basic mechanism of building a UUP in BEA WebLogic Portal. You can create a user profile by aggregating any number of legacy data sources by deploying and registering appropriate EntityPropertyManager EJBs. And you can – among other things – provide personalized content and fine-grained access control within your portal application.

Of course, in a real-world project there are several other important considerations for the actual design of these EJBs. For example, your back-end data sources are not as available and reliable as you need them to be to satisfy your requirements. In this case, you might create a local persistent cache. This cache may be rather short lived and may only serve as a fallback data store in case the actual systems that provide your data are off line. You may also use the BEA caching library to provide better control for the cached data in your EntityPropertyManager.

Considering future developments, you should also have a look at another tool from BEA. WebLogic Liquid Data (see article in ***WLDJ***, Vol. 2, issue 4) is specifically designed to aggregate data from diverse data sources. Plugged into WebLogic Portal it will make the process of implementing user profiles significantly easier and a lot faster.

References

- ***BEA WebLogic Portal Documentation: Implementing User Profiles:*** http://edocs.bea.com/wlp/docs70/dev/usrgrp.htm#998993
- ***BEA WebLogic Portal Documentation: Portal Content Management:*** http://edocs.bea.com/wlp/docs70/dev/conmgmt.htm#1018613

Informatica

www.informatica.com

The Performance Dragnet

TRUE STORIES OF JAVA APPLICATION PERFORMANCE

BY **DAVE MARTIN**

AUTHOR BIO...

Dave Martin is a system engineer for Wily Technology. He has worked extensively with dozens of enterprise customers to identify and correct Java application performance issues. A former software developer for Wily technology, Dave has a comprehensive knowledge of J2EE architecture and software systems design and implementation.

CONTACT...

dmartin@wilytech.com

For any organization, the two fundamental requirements for ensuring that enterprise applications meet high standards for performance are the ability to monitor the application with near-zero overhead and the ability to determine the root cause of problems quickly when they arise, regardless of whether the application is in QA, staging, or deployment.

The stories you are about to read are true. The names have been changed to protect the (sometimes not-so) innocent.

This article is the documented drama of actual Java performance problems. It begins with a summary of Wily's Enterprise Java Application Management solution and the role it plays in pre- and post-deployment performance analysis. Then I'll look at true stories of performance gone bad. You will learn – the easy way – why skipping Java application management doesn't pay.

The Wily 4 Solution

As a healthy organization matures, it will naturally develop a set of strategies and processes for eliminating the amount of turmoil it has experienced in fighting performance problems associated with its mission-critical Java applications. These organizations also understand the heavy cost associated with poor application performance: the returns on IT investments, the risks to the organization's revenue stream, and ultimate customer satisfaction.

The Wily 4 solution from Wily Technology allows enterprises to monitor, improve, and manage enterprise Java applications in each stage of the application life cycle. It provides a common language that each department within an IT organization can use to quickly identify and fix Java application performance problems when they arise.

With Wily 4, IT teams can monitor the business function of their production applications 24/7, isolating performance bottlenecks in fund trans-

fers, bill payments, purchase of goods, and other core use cases – all of which can be mapped, without source code changes, onto the underlying methods of the servlets, JSPs, EJBs, and custom code that implement them. Customizable dashboards and a wide variety of options for integration make Wily 4 a perfect complement to popular systems management solutions. It allows organizations to publish Introscope Alerts with information about the availability of the production application's customer functions right into their existing operations' problem resolution pipeline (see Figure 1)

Key elements of the Wily 4 solution include Introscope, its flagship Java application monitor; Introscope LeakHunter for identifying the potential sources of memory leaks in production systems; and Introscope Transaction Tracer for solving performance problems associated with individual transactions. The Wily 4 solution also includes Introscope SQL Agent and Introscope PowerPacks for identifying application performance problems associated with the connections between Java applications and supporting back-end systems. Together, these solutions offer a "whole application" view into the entire Java environment.

The Performance Dragnet: Setting and Achieving Performance Goals

As a best practice, organizations use Wily's Introscope to benchmark and record the responsiveness of an application's key components to achieve performance goals. Successful development and deployment of enterprise Java applications should include continual baseline measurements of core use cases even as they are developed. As developers add and commit changes to an application, these baselines provide hard, statistical means by which each subsequent build of the application can be evaluated. If you fail to watch carefully through each step of the development process, you can end up with bugs or, far worse, architectural bottlenecks that are far more costly to fix in staging or production.

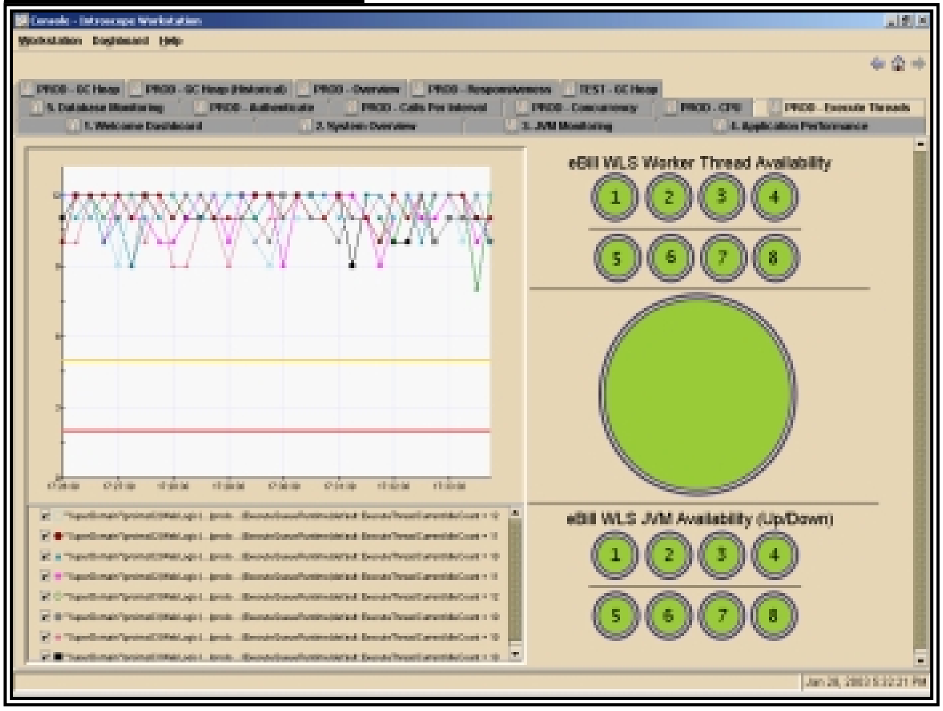
Like a good detective, a good baseline or benchmark just keeps asking the same questions until it gets the answers it wants. This kind of process encourages a "measure twice, cut once" approach that has proven itself time and again as the most effective means of improving software performance.

To successfully load all the key use cases of an application, the application must be fully functional; a good set of performance load-generation scripts can double as excellent sanity checks for functional availability of core application features.

Once the application is under load, organizations need to be able to continuously monitor the application's performance – at a feature level – to identify bottlenecks, especially during the first few months of production. Wily's Introscope can abstract the details of your Java metrics into views and reports that actually describe the application's features. In the ubiquitous PetStore example, this would be metrics about purchase of goods, return of receipt, view of catalog items, inventory search, and so on. Taking this approach, you're not just tuning a component because it is slow, but because you have observed that it makes the time for return of a customer's receipt take unacceptably long. Without this kind of a view, you risk wasting valuable development time on tuning features that may not be as important to the application's overall success.

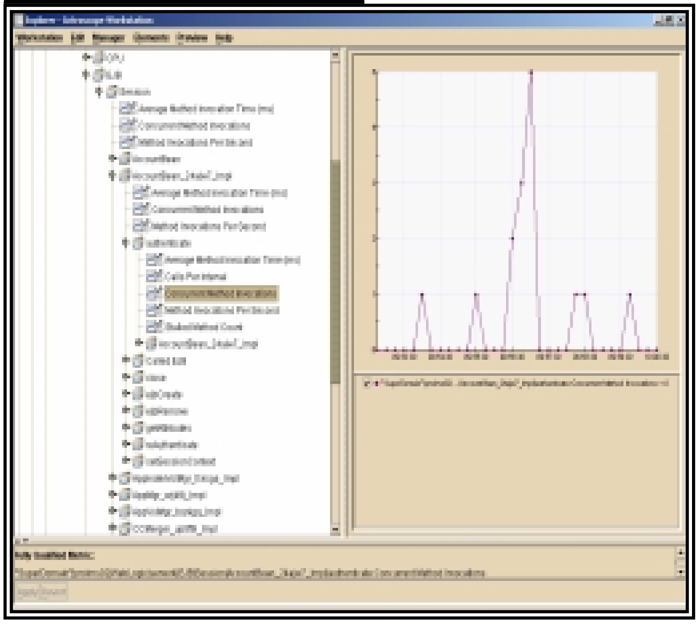
Wily's Introscope reports the call volume, mean, minimum, and maximum response time of all the key components of the customer's applications. Introscope can also measure concurrency (a measure of the

FIGURE 1



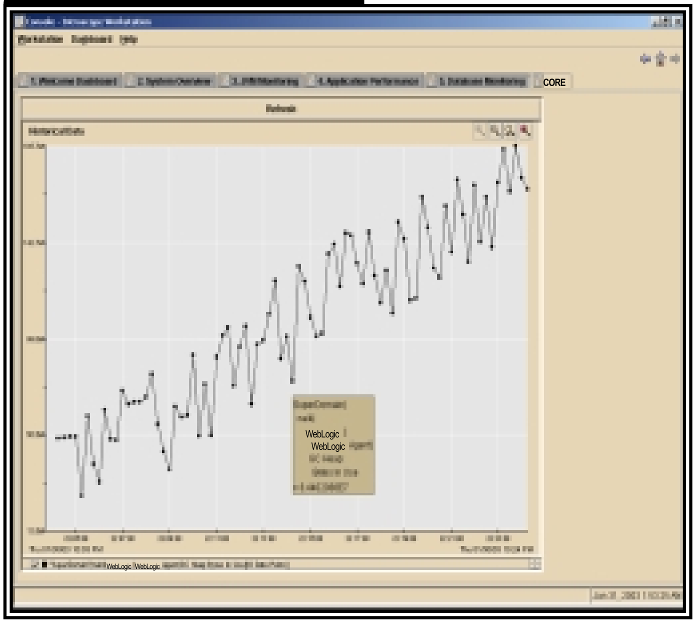
Introscope offers custom views into the whole Java application environment.

FIGURE 2



Introscope shows threads backing up on the authenticate method of an EJB.

FIGURE 3



Introscope shows a GC Heap Bytes-in-Use pattern indicating a memory leak.

threads busy in each method or component of business logic), memory statistics, file and socket i/o statistics, pooled resource utilization statistics (e.g., MQ connections), environmental statistics, and more.

In the prototypical J2EE application, the controller servlet, model EJBs, back-end connector APIs (e.g., JDBC, MQ Series, etc.), and view JSPs would be measured. Organizations often take the additional, simple step of customizing Introscope so that they can monitor, without source code changes, their own proprietary business logic implementation or back-end connectors. For example, Introscope has monitored SABRE transactions during work at major airlines, legacy telecommunications company mainframe callbacks, and a myriad of other such custom back-end systems and business logic.

Data from the application can be recorded to CSV flat files, the most popular databases, or integrated directly into many load-generation tools. Each customer chooses the means of data recording and analysis that best suits their needs. Data written to flat files is automatically archived at configurable file size limits, making for easy scripted automation of both reporting and cleanup. Some of the most common databases also have example data archiving scripts available.

The remainder of this article focuses on a number of real-world instances in which Introscope was used.

"The Big Sleep"

It was a Monday in Philadelphia, overcast and cold. I was working the day shift. The customer was an insurance company with an application in staging.

The application we were monitoring, we'll call it AUDIT, was a Web front end to a legacy green-screen application. Underwriters for the customer use the application to create new policy quotes. AUDIT uses CICS primarily for its back-end data, issuing incoming transactions against the same legacy stored procedures it has for the last 30 years, then parsing them up into JSPs. Pages were designed to be loaded in 1–2 seconds, but requests were taking 7–8 seconds each click. Data from CICS indicated that all requests were being honored quickly. We needed to get to the bottom of the situation fast.

Wily Introscope Custom Tracers implemented on the application's data object parsing procedures showed response times of exactly 100 milliseconds for each call, with 50–60 calls being made per transaction. Quick investigation into the data object construction revealed a developer's placeholder logic. Each object required a unique ID. A 100-millisecond sleep was programmed into the constructor in order to ensure a unique timestamp from each subsequent call to the system clock.

Epilogue: The developer who controlled the knob for remotely adjusting the sleep time was never found.

"The Adventure of the Speckled Bean"

It was a Tuesday afternoon in Texas. I was at a customer who had already logged several months trying to solve their application performance problem. When I asked Dr. Watson's opinion, he replied with a core dump.

In production, the STORE application was losing about 30% of customer purchases due to slow response times. Some requests inexplicably took longer than two minutes, then timed out. Introscope quickly isolated the performance bottleneck to a method that was being used to calculate sales tax for each item of the purchase. A single line of Introscope configuration was enough to trace the business logic interface implemented by all STORE's business logic.

Further investigation revealed that the method used Runtime.exec to invoke a new JVM. Out-of-process, this new JVM calculated a single item's sales tax, printing its results to system out. The application server patiently waited for the printout, then did the same thing again for each item purchased. Under modest load, as many as 50–60 JVMs were being started up and shut down simultaneously, requiring 20–40 seconds of wait time for each. The transactions being lost were those with the most purchased items.

Why? The person tasked with using the EJB interface included with the third-party sales tax calculation software had been unable to load it into the server's context.

Hewlett-Packard

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

Frustrated, he fell back on using the software's example code. This was easily reworked into a script that took appropriate command-line parameters and generated the desired result on system out.

Epilogue: The suspect pleaded Perl and was dismissed. Case closed.

"And Then There Were None (JDBC Cursors)"

It's Wednesday and I am steps away from the historic Boston Common. The customer was consistently running out of JDBC cursors and MQ Series connections during the course of their lengthy load tests. Wily's Introscope monitored the allocate and de-allocate semantics of these resources. With Introscope's Blame feature, the team was able to quickly isolate the guilty EJB: under heavy load the application was throwing exceptions that skipped the usual close calls, which should have been enclosed in finally blocks to ensure release.

Epilogue: The missing "close" is now safely behind finally block bars.

"Dial S for Synchronized"

Thursday, and my dogs are barking. I drink my fifth cup of joe and head out the door.

Wily's Introscope measured the average responsiveness of transactions with a proprietary back-end system as 10 seconds in length. In a load test of 10 concurrent users, however, the average response times of transactions that used this back-end system were in the 90–100 second range. This case was getting more and more interesting. A concurrency metric on the call to the back-end system quickly isolated the discrepancy: nine concurrent user threads were sitting on a call to the back-end system while one thread actually transacted work.

Problems such as a misplaced synchronized keyword in the class signature instead of the method signature, a resource pool with a single instance where 10 should have been available, or a remotely accessed server that becomes unavailable in fits and bursts can and have been caught with these Wily concurrency metrics (see Figure 2).

In this case, the application was simply incapable of dealing with unexpected failures of key back-end systems, taking minutes or even hours to time out a call that customers were expecting to come back in seconds. Concurrency is a quick and easy way to identify such spots in the application, making it simple to handle similar failures in production more gracefully.

Epilogue: The application and its owners are now doing hard time in a production facility.

"The Maltese ResultSet"

It's Friday and I'm in Atlanta. Peachtree was everywhere: the street, not so much the tree.

This customer had a problem with large database result sets. In neglecting to architect a strategy by which search queries could be limited and doled out in reasonable increments, they created a "pig in the python" memory problem for their application server. The wrong kind of search or lookup resulted in a JVM gone wild with thrash on the GC Heap. The application server struggled to page-in the result set as it was handed off by the database. Utter destruction of responsiveness for all other transactions with the application was the ugliest side effect.

Wily's Introscope SQL Agent immediately diagnosed the problem by showing the difference between the database queries' response time and "roundtrip" response time. The latter metric included any clock time spent before calling close on the ResultSet, while the former included only the time taken to recover an answer from the database. Roundtrip response times showed extremely large discrepancies from their reciprocal metric. This metric was, of course, recoverable only from inside the application server.

Epilogue: The development team made the necessary corrections and the application is now a fully reformed and contributing member of society.

"Murder on the JDBC Express"

I volunteered to work a Saturday up in San Francisco. After that last customer visit to Vegas, I needed the overtime.

Little did I know that my JDBC woes were far from over. The customer had generally figured out how to tune the response time of their key database interactions – JDBC statements were measured with a rough mean of 10 milliseconds, and most were in the 1 millisecond range.

What about the transaction that fell out of that range? I slapped Introscope Transaction Tracer on the application and recorded a host of 30-second transactions during the standard load test. Previously, no amount of profiling had helped the customer to find the "guilty" method. The Transaction Trace call graphs, automatically weeded of all extraneous data by Introscope, allowed the customer to see the forest for all of its trees: nearly 4,000 JDBC statements were being issued by a single transaction. Even with these fast responses from the database, so many queries are enough to sink a click.

Analysis of the Transaction Trace reveals a single validation method responsible for six JDBC statements for each call – it's called enough to account for almost 40% of the JDBC statements in each transaction. With a trivial number of changes from the development team – less than a day's work – these validation checks were implemented with a caching strategy. CPU burn and transaction time plunged; throughput leapt.

Epilogue: The database made a full recovery but the development team has since been sentenced to Swing.

"The Vector Who Knew Too Much"

It was Saturday night, the end of a long week. The city lights filter through the fog rolling off the bay. The phone rings just as I'm tilting a tall glass of rye.

The customer had a J2EE-based trading infrastructure with calls made directly into their EJB remote interfaces. Months ago, they swapped their JMS implementation and were now experiencing intense CPU burn and reduction in throughput. The "GC Heap Bytes in Use" were rocketing with each subsequent transaction, resulting in an out-of-memory exception. The team rounded up all the usual profiling tools but none of them could identify the source of the leak (see Figure 3.)

Wily's Introscope LeakHunter installed on the applications in minutes and immediately flagged anomalous growth in a single Vector – one of the 45,000 data structures used by the application. The Vector in question contained transaction identifiers, meant to be cached only temporarily for quicker rollback. Instead, the Vector grew one-to-one with the number of transactions made against the application. The new JMS implementation answered questions about its transactions in a slightly different way, leading to a different, unexpected result when "contain" was called on the Vector. The unexpected false answers when "contain" was called resulted in new items being added with every transaction.

Epilogue: The offending Vector now goes by the name of Victor and is living in Florida under the Witness Protection Program.

After a long goodbye, I headed off into the night, not knowing where my next case would take me.

SaturdaySessions

We understand the pressures of work and how difficult it can be to get time off. That is why we have designed this workshop to be held in one day and, as a special bonus, on the weekend, so no days off from work. **Your boss will be happy!**

Alan Williamson
JDJ Editor-in-Chief

JDJ Workshop
with Alan Williamson

M	T	W	T	F	S	S

Coming to you...

April:	May:	June:
NEW YORK	BOSTON	ATLANTA
WASHINGTON, DC	TORONTO	RALEIGH

Performance > Efficiency > Reliability

This one-day intensive workshop is designed for developers who wish to increase the efficiency and reliability of their code development.

1) The day will begin by looking at the various hints and tips you can utilize at the code level to improve the quality and reduce the number of bugs you have to contend with.

2) The next part will look at Apache's Ant and how you can use this freely available tool for your own development, irrespective of your IDE.

3) Last, and most important, as the old saying goes: "You can never do enough testing." This session will look at JUnit and show you how to start building test harnesses for your code so you can begin your testing strategy.

>Performance

Java is a powerful language. While it offers a rich array of tools, the fundamentals mustn't be overlooked. Improving your code at the core layer will result in great improvements in efficiency and produce (hopefully) less bugs. We'll look at the do's and don'ts of programming and learn many hints and tips that will accelerate your Java coding.

>Efficiency with Ant

Apache's Ant is a powerful scripting tool that enables developers to define and execute routine software development tasks using the simplicity and extensibility of XML. Ant provides a comprehensive mechanism for managing software development projects, including compilation, deployment, testing, and execution.
In addition, it is compatible with any IDE or operating system.

> Reliability with JUnit

A critical measure of the success of software is whether or not it executes properly. Equally important, however, is whether that software does what it was intended to do. JUnit is an open-source testing framework that provides a simple way for developers to define how their software should work. JUnit then provides test runners that process your intentions and verify that your code performs as intended. The result is software that not only works, but works in the correct way.

What you will receive...

✓ *INTENSIVE ONE-DAY SESSION*

✓ *DETAILED COURSE NOTES AND EXCLUSIVE ONLINE RESOURCES*

✓ *JDJ CD ARCHIVE*

ONLY $295*
*JDJ subscribers
($395 non-subscribers)
*GROUP DISCOUNTS AVAILABLE

To Register

www.sys-con.com/education

Call 201 802-3058

SPONSORED BY
JAVA DEVELOPER'S JOURNAL

PRODUCED BY
SYS-CON EVENTS

PRESENTED BY
i-TECHNOLOGY EDUCATION

Together ControlCenter Accelerator for WebLogic Workshop

EXPANDING ON A NARROW FEATURE SET

Reviewed by Steve Buzzard

Product contact info:
Borland Software Corporation
100 Enterprise Way
Scotts Valley, CA 95066-3249 USA
Ph: (831) 431-1000

Customer Service:
customer_service@borland.com

Corporate and Government Sales West/Central:
1 (800) 632-2864
East: 1 (800) 662-2667

Platform Requirements
The ControlCenter Accelerator for Workshop requires ControlCenter version 6.0.1 and integrates with WebLogic Workshop 7.0. It is supported on Microsoft Windows (NT, 2K, XP), Linux, and Solaris.

This Accelerator requires that the following software is installed on your system:

- Together ControlCenter 6.0.1 or higher
- BEA WebLogic Server 7.0 or higher
- BEA WebLogic Workshop 1.0 or higher

Download
www.togethersoft.com/developers/integrations/beawlw_download.jsp?c=205

Pricing
The price for the Accelerator is $500.00.

Before I dig into this review, I should let it be known that I have a lengthy background in, and preference for, command-line tools. Scripting is my thing. I love tools like Ant, Cactus, XDoclet, and EJBGen. I get frustrated when I'm dealing with tools that make it hard to peek behind the scenes to see what's going on. Generally, I don't get excited about IDEs. I certainly understand and respect their value (especially when dealing with something as complex as J2EE design, development, and deployment). I take advantage of whatever IDE my current client has standardized on to perform such tasks as remote debugging, EJB entity bean/database mapping, and so on.

I'm a fan of the BEA Weblogic Workshop 7.0 IDE's look and feel. It is amazingly lightweight and really intuitive. As nice as the Workshop IDE was, though, it was narrowly focused and stand-alone (you have to go elsewhere to model and develop your server-side components).

This brings us to this review: the ControlCenter Accelerator for BEA WebLogic Workshop. Accelerators are ControlCenter plug-ins that integrate additional features into the product. For those not in the know, I'll briefly describe the WebLogic Workshop technology, and then get into the review itself .

BEA WebLogic Workshop

WebLogic Workshop simplifies the development, deployment, and debugging of Web services. Workshop also provides transparent message buffering and stateful Web services with conversational capabilities. Workshop developers are provided with several out-of-the-box "controls" that allow you to easily expose any number of back-end J2EE components as Web services. It uses a "javadoc"-style metadata facility to describe the Web services without necessarily having to know the Java programming language. The metadata is used to generate the appropriate Java classes that implement the Web services. Workshop consists of two separate implementations: the server-side components embedded into WebLogic Server and the associated IDE. The Accelerator interacts with the server-side components and integrates the functionality of the Workshop IDE within ControlCenter.

Installation/ Configuration

Installation is a breeze. Just execute the Accelerator install program and go (ensure that you already have ControlCenter 6.0.1 installed first).

Once installed, configuration is simply a matter of going into ControlCenter's Tools->Options ->Default Level, selecting the Web services node in the options hierarchy, and then BEA WebLogic Workshop within this node. Enter the information about your specific WebLogic Workshop Domain and save it. If you don't yet have a Workshop Domain set up, you can use the WebLogic Platform's domain wizard to do this (selecting the Workshop domain type). Figure 1 shows a sample Accelerator configuration.

Features

The following list represents the major features of the ControlCenter Workshop Accelerator. The example that follows illustrates these features while developing a simple EJB and exposing it through a Workshop Web service.

- ***WebLogic Workshop Diagrams:*** Provide new Web service diagrams. Add Web services and controls, and connect them visually while staying synchronized with the source code.
- ***JWS and CTRL editing:*** Because JWS files and CTRL files are Java class files, developers can take advan-

tage of ControlCenter's powerful editing features, which include code sense, syntax highlighting, and macros.
- ***Control creation:*** Create controls from Web services, whether they are JWS files or defined by WSDL (Web Service Definition Language). Create database controls from ER diagrams, or EJB controls from existing EJBs.
- ***Deployment:*** Deploy and test with the click of a button.

Accelerator Example

The following example demonstrates some of the capabilities of the Accelerator. I'll create a "Hello World" session EJB and deploy it to the WebLogic server. In addition, I'll access the EJB in a Web service using a Workshop EJB control.

Step 1: Setting Up the Project

We'll set up the project by creating a Web service diagram.
- Click the New Diagram button on the horizontal menu bar of the Designer Pane to open the New Diagram dialog.
- Select the Together tab in the dialog and then choose the Web service diagram type.
- For Diagram name, enter "HelloWorldServiceDiagram" and then click OK.
- Click the Start WebLogic Server button on the horizontal menu to start the server.

Step 2: Modeling the EJB

Now you'll design the HelloWorld session EJB. In addition, you'll add the EJB to an assembler diagram so that you can deploy the EJB. (An assembler diagram represents a .jar file that contains the EJB.)
- From the Designer pane, select the tab for the <default> diagram.
- Click the Session EJB button on the vertical toolbar. Click on the diagram where you want to place the EJB.
- Rename the EJB to "HelloWorldSessionBean".
- Right-click the EJB and choose New | Business Method.
- In the Designer pane, rename the method as follows:

```
+sayHello:String
```

- In the Editor pane, add the following return statement for the method:

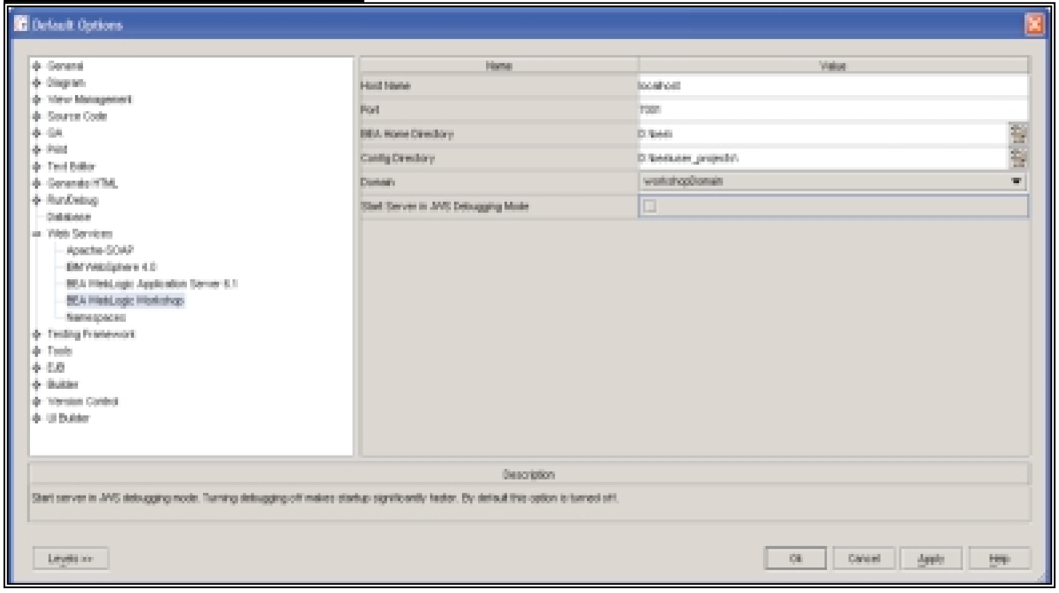
```
public String sayHello(){
return "Hello World";
}
```

- Click the New Diagram button on the horizontal menu bar of the Designer Pane. The New Diagram dialog opens.
- Select the Together tab in the dialog and then choose EJB Assembler.
- For Diagram name, enter "helloWorldJAR" and click OK.
- Make sure that helloWorldJAR is open in the Designer pane. Go to the Explorer pane, right-click on the HelloWorldSessionBean, and choose Add as Shortcut. The EJB appears in the assembler diagram.

Step 3: Modeling the Web Service and EJB Control

We'll create an EJB control for the session EJB created in the previous step. In addition, we'll model a Web service that instantiates the EJB control.
- From the Designer pane, select the tab for HelloWorldServiceDiagram.
- Click the Generate EJB Control from EJB existing in Model button on the horizontal menu bar. The Select EJB(s) from deployment diagram dialog opens.
- Click the file chooser button and expand the Model node.
- Select the helloWorldJAR node and click OK.
- Click OK to add HelloWorldSessionBean Control to your Web service.

FIGURE 1



Name	Value
Host Name	localhost
Port	7001
BEA Home Directory	D:\bea\
Config Directory	D:\bea\user_projects\
Domain	workshopDomain
Start Server in JWS Debugging Mode	☐

Description

Start server in JWS debugging mode. Turning debugging off makes startup significantly faster. By default this option is turned off.

Accelerator Configuration

- Click the Web service button on the vertical toolbar and then click on the diagram.
- Rename the Web service "HelloWorldService".
- Use the Web Service Association Link button on the vertical toolbar to connect the Web service with the EJB control. Note that the link also generates an instance of the EJB control in the Web service named "myHelloWorldSessionBean Control".
- Rename the instance variable "myEJB". Figure 2 shows the updated HelloWorld Service Diagram

Step 4: Building the Project

We'll add functionality to our Web service so that a client user on the Web can see the result of calling the session EJB's sayHello() method. Specifically, we will add a method that can access the EJB using the EJB control.

- From the HelloWorldServiceDiagram, select HelloWorldService and then click the Add Method button on the horizontal menu bar. This adds a method called method1() to the Web service.
- In the Designer pane, rename the method signature as follows:

```
+callMyEJB():String
```

- In the Editor pane, code the implementation for this method as follows:

```
/** @jws:operation */
public String callMyEJB()throws
java.rmi.RemoteException {
  return myEJB.sayHello();
}
```

Note: When we used the Add Method button in the previous step, Together ControlCenter automatically inserted the /** @jws:operation */ tag.

Step 5: Deploying the Project

We'll deploy our project by testing the Web service in a Web browser.

- Select HelloWorldService in the diagram and click the Test Web Service button on the horizontal menu bar of the Designer pane. The Deployment Expert for the helloWorldJAR diagram appears.
- Uncheck the Open XML editor for the generated Deployment Descriptor(s) option.
- Check the Hot Deploy to server option.
- Accept the remainder of the default settings for the expert, clicking Next to continue and then click Finish. After deployment, a Web browser opens.
- In the Web browser, select the Test Form tab.
- Click the callMyEJB button to call the session EJB.
- Verify the result of the operation. The EJB should return the string "Hello World".

Summary

The really nice thing about this product (and, really, its main reason for being) is that it takes the powerful, but relatively narrowly focused and standalone, feature set of the WebLogic Workshop 7.0 IDE and integrates it into ControlCenter's comprehensive J2EE design, development, deploy environment. This enables a developer to intermix Workshop Web services development with that of EJBs, servlets, JSPs, and Apache SOAP Web services, among others, while utilizing ControlCenter's modeling, refactoring, design pattern support, and documentation-generation capabilities.

FIGURE 2

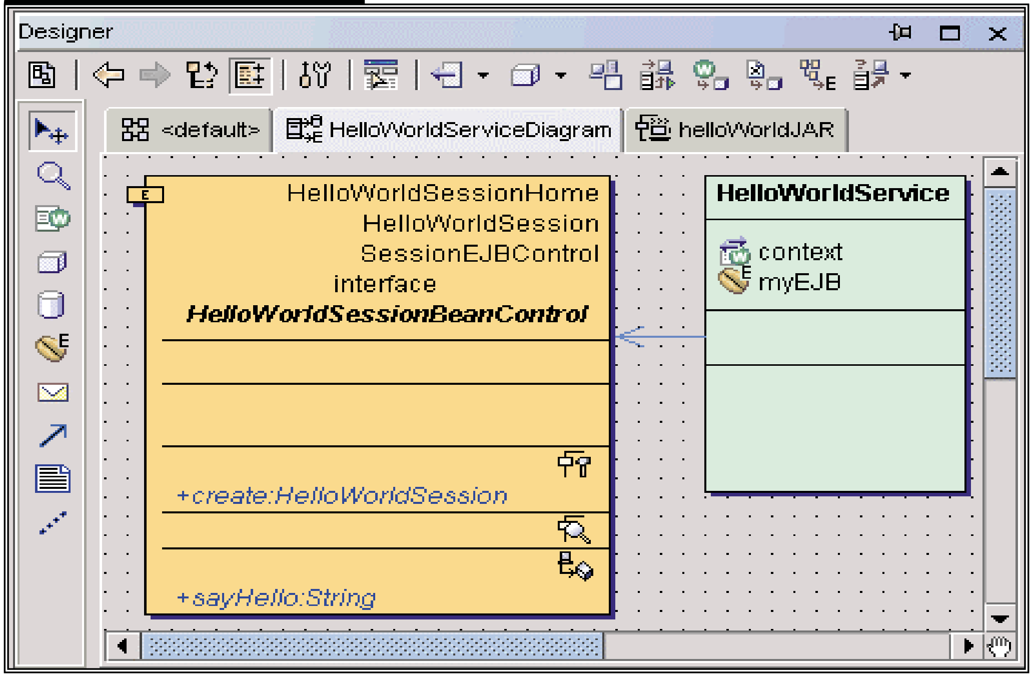


Diagram for HelloWorldSession Service

AUTHOR BIO

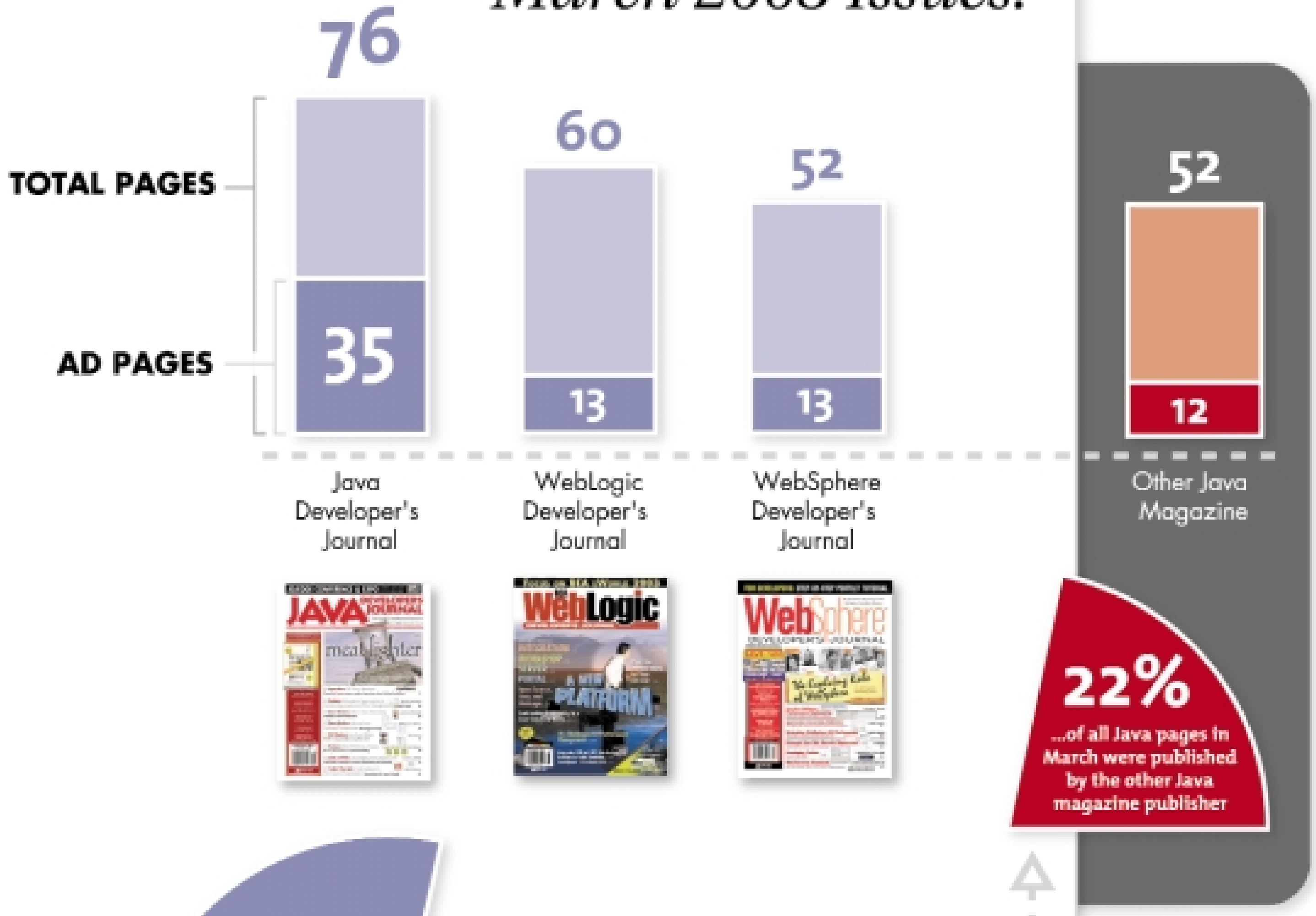
Steve Buzzard is a J2EE application architect with Anexinet Corporation (www.anexinet.com), a leading Philadelphia-based consulting firm. Steve has over 17 years of experience in professional software development and has been working almost exclusively with the WebLogic technology stack since late 1998.

CONTACT...

sbuzzard@anexinet.com

Java Served Here...

March 2003 Issues:



22% ...of all Java pages in March were published by the other Java magazine publisher

78% ...of all Java pages in March were served by SYS-CON Media

SYS-CON MEDIA

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

JAVA DEVELOPER'S JOURNAL

JMSML: XML-Based Mark-Up Language

CUTTING BACK ON THE DETAILS OF PROGRAMMING, MONITORING, AND TESTING

BY **KATHIRAVAN SENGODAN**

JMSML is a mark-up language designed and developed to make Java Message Service (JMS) and Java Management Extensions (JMX) programming easy by hiding all the JMS and JMX Java API complexity behind a few easy-to-use XML tags.

AUTHOR BIO

Kathiravan Sengodan has more than 11 years of experience in the software industry and is currently a software engineer on the BEA WebLogic JMS/Messaging team.

CONTACT...

kats@bea.com

JMS Applications: A Background

A typical JMS application development process involves configuring and managing the JMS server components, like JMSConnectionFactory and JMSDestination, on the JMS Provider application server, writing JMS application clients that will make use of these administered server-side objects to do the Java messaging.

JMS application clients are written in Java using the JMS API, and are categorized into two types of programs:

- ***Producers:*** For creating various JMS Message types (like Text, XML, Object, Stream, Bytes) and sending them to the JMS destinations (Queues and Topics)
- ***Consumers:*** For receiving messages from the JMS destinations (Queues and Topics), both synchronously and asynchronously

The JMS API enables both producers and consumers to utilize various qualities of services (QOS) that are provided by the underlying JMS implementation and by the JMS providers (like transaction and acknowledgment).

Writing producers and consumers involves following some specific steps, in a specific order, while using the JMS API. This process has to be repeated for every single producer and consumer that is written to do Java messaging.

The JMSML Approach

The JMSML approach to writing JMS application clients is accomplished without writing Java code. Instead, JMS API complexity is abstracted into a few XML tags that are easy to use and remember. Using JMSML, you can create simple reusable JMS components, like a "Sender", "Receiver", "Publisher", and "Subscriber". In addition, JMSML makes administration of a JMS server very simple by using XML tags, thus eliminating the JMX Java API complexity. JMSML supports all the JMX operations that are needed to do dynamic configuration, management, and runtime monitoring of a JMS server.

Finally, JMSML simplifies testing by automating verification of operation and monitoring output (see Figure 1).

What Does a Java JMS Program Look Like?

Listing 1 shows a Java program using the JMS API to send a text message, "Hello World", to a JMS queue named "exampleQueue", using a JMSConnectionFactory named "QueueConnectionFactory" via a nontransacted, autoacknowledge JMSSession.

What Does a JMSML Program Look Like?

The following 9 lines of a similar XML program, QueueSend.xml, are equivalent to the 74 lines of QueueSend.java in Listing 1. Note that all other Java JMS API details are taken care by JMSML by means of having default attribute values.

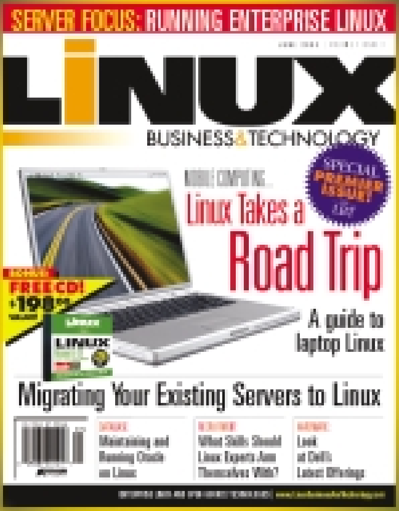
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jmsml PUBLIC "-//BEA Systems Inc//DTD JMSML
    Version 1.0 JMSML//EN" "jmsml.dtd">
  <jmsml>
    <operation Class="message" Type="Text"
        Name="MyQueueSender"
      Factory="QueueConnectionFactory"
      Queue="exampleQueue"
      Body="Hello World"
    >Send</operation>
  </jmsml>
```

In this case, JMSML will create QueueSender with a nontransacted and autoacknowledge JMSSession. Once this operation is executed, then the "MyQueueSender" object is still available for reuse later, until the end of the program execution. That is, "MyQueueSender" can be reused to send different messages to different queues.

Premiering June 2003 at JavaOne

Sun's 2003 Worldwide Java Developer Conference

www.sys-con.com



Millions of Linux Users One Magazine

Linux Business & Technology

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

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

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

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

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

Regular features will include:

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

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE $71.76

YOU PAY ONLY $49.99

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM
OR CALL
1-888-303-5282

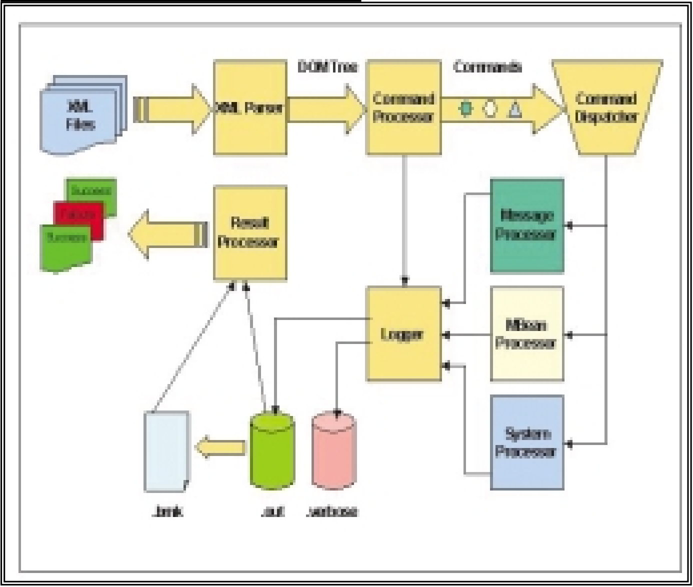
LINUX EDGE conference&expo

June 3–5	LONDON
June 24–26	BERLIN
September	HONG KONG
October	CALIFORNIA

FOR ADVERTISING INFORMATION:
CALL 201 802.3020 OR
VISIT WWW.SYS-CON.COM

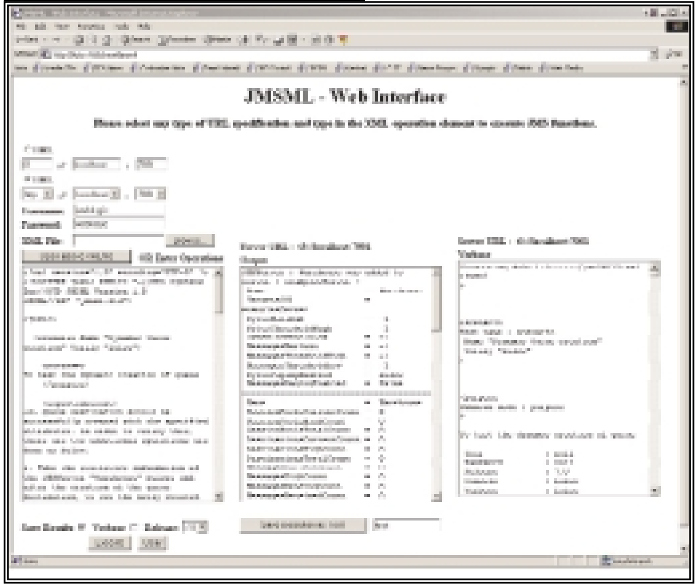
SYS-CON MEDIA
The World's Leading i-Technology Publisher

FIGURE 1



JMSML Extensible Engine architecture

FIGURE 2



JMSML Web browser interface

JMSML Applications

From the previous code comparison, you can easily figure out that JMSML is simple to learn and use yet very powerful in terms of its functionality. JMSML can be used to do the following tasks in the WebLogic JMS subsystem.

- As an administration and monitoring tool for the WebLogic JMS subsystem
- As a prototyping tool for WebLogic JMS applications
- As a testing tool for the WebLogic JMS Subsystem

Using JMSML as a Testing Tool

Once JMS testing scenarios are identified, they can easily be translated into JMSML operations and then grouped as testing scenarios using <scenario> with the *Verify* attribute set to "false". This test program can be executed using one of the JMSML execution models. After successful execution, an output file is created with the same name as the input JMSML program and with a .out extension. It is verified manually for correctness and then saved as a benchmark file (again in the same name as the input JMSML program but with a .bmk extension).

Later, whenever the same JMSML program is executed, with the *Verify* attribute of the <scenario> element set to "true", JMSML treats that input JMSML program as a test, and the output file is automatically compared with the corresponding benchmark file for test verification. If the benchmark and the output file matches, then the test is declared as passed. Otherwise, it will be declared as failed.

Example

Test the dynamic creation, management, and monitoring of the JMS administered objects and the message send, receive features of JMS.

Testing Scenario: Create a JMSServer named "Warehouse" with all the default attribute values and deploy it on a WebLogic Server instance named "examplesServer". Create a queue destination named "orderqueue" with all the attribute values explicitly defined, add it to the JMSServer named "Warehouse", into JNDI as "jms.queue.order", send/receive messages to this queue, and at the end delete both the queue "orderqueue" destination and the "Warehouse" JMSServer.

Listing 2 is the JMSML representation of the Testing Scenario defined above.

Once a JMSML program is written, it can be run against a WebLogic Server instance. Currently, JMSML provides three different ways to execute a program they are;

1. Stand-alone commandline client
2. Web browser interface for remote execution
3. IDE (prototype)

```
java com.bea.jmsml.commandline.JMSMLClient \
-protocol t3 \
-host localhost \
-port 7001 \
  -username weblogic \
-password weblogic \
  -filename jmstest.xml
```

This command line assumes the following about the execution environment:

- An instance of WebLogic Server named "examplesServer" is running on "localhost" and is listening on the port '7001" with the "t3" protocol enabled.
- WebLogic security is set up such that a user named "weblogic" with a password of "weblogic" is configured with permissions for creating and accessing JMS server components.
- A valid JMS connection factory is deployed on the "examplesServer" and is bound to JNDI as "weblogic.examples.jms.QueueConnectionFactory".
- The test program shown in Listing 3 is saved in the current directory along with the "jmsml.dtd" file.

After the successful execution of the command, the following message will be printed out to the stdout.

```
*** jmstest processed *** \n*** Please see
./jmstest.out for the results ***
```

At this time, an output file named "jmstest.out" will exist in the current directory, containing all the operations execution results, as shown in Listing 3. In this output sample, the contents are color-coded to identify the operation category, as follows:

A LIMITED TIME SAVINGS OFFER FROM SYS-CON Media

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

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

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

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

☐ **Linux Business & Technology**

U.S. - Two Years (24) Cover: $143	You Pay: $79.99 /	Save: $63 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
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

☐ **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

☐ **WebSphere Developer's Journal**

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

☐ **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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

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
- Canada/Mexico – $245.00
- International – $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
- Canada/Mexico – $555.00
- International – $710.00

9-Pack

Pick 9 of our magazines and save up to $400^00

Pay only $495 for a 1 year subscription plus 3 FREE CDs

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

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

SYS-CON MEDIA

- ***Brown:*** JMS Receive (successful Send has no output in normal mode)
- ***Red:*** JMX Add, Remove
- ***Blue:*** JMX List (config information)
- ***Green:*** JMX List (runtime statistics)

Once this output is manually verified for correctness, you can easily make it a valid, reusable WebLogic JMS test case by doing two things:

1. Rename the "jmstest.out" file to "jmstest.bmk"
2. Edit the "jmstest.xml" file by changing the Verify attribute value to "true" in the <scenario> element.

The next time the same command line is executed, JMSML treats the "jmstest.xml" as a test case and prints out the test pass/fail result to the stdout.

That's it! Without writing a Java program, we have quickly written a complete test case for BEA WebLogic JMS and JMX features using JMSML. This is a working example and is packaged with JMSML download.

Resource

The JMSML product binary, along with the technical white paper, can be downloaded from: <http://dev2dev.bea.com/resource library/whitepapers/JMSML_WhitePaper.jsp> http://dev2dev.bea.com/resourcelibrary/whit epapers/JMSML_WhitePaper.jsp

Listing 1: QueueSend.java

```java
import java.util.*;
import javax.naming.*;
import javax.jms.*;

public class QueueSend
{
public final static String
         JNDI_FACTORY="weblogic.jndi.WLInitialContextFactory";
public final static String
          JMS_FACTORY="QueueConnectionFactory";
   public final static String QUEUE="exampleQueue";

  private QueueConnectionFactory qconFactory;
  private QueueConnection qcon;
  private QueueSession qsession;
  private QueueSender qsender;
  private Queue queue;
  private TextMessage msg;

  public void init(Context ctx, String queueName)
       throws NamingException, JMSException
  {
 qconFactory = (QueueConnectionFactory)
                 ctx.lookup(JMS_FACTORY);
    qcon = qconFactory.createQueueConnection();
 qsession = qcon.createQueueSession(false,
                                     Session.AUTO_ACKNOWLEDGE);
    queue = (Queue) ctx.lookup(queueName);
    qsender = qsession.createSender(queue);
    msg = qsession.createTextMessage();
    qcon.start();
 }

  public void send(String message)
       throws JMSException
  {
    msg.setText(message);
    qsender.send(msg);
  }

  public void close()
       throws JMSException
  {
    qsender.close();
    qsession.close();
    qcon.close();
 }

  /** main() method.
   *
   * @param args WebLogic Server URL
   * @exception Exception if operation fails
   */
  public static void main(String[] args)
      throws Exception
  {
    if (args.length != 1) {
      System.out.println("Usage: java QueueSend WebLogicURL");
      return;
    }
    InitialContext ic = getInitialContext(args[0]);
    QueueSend qs = new QueueSend();
    qs.init(ic, QUEUE);
    qs.send("Hello World")
    qs.close();
  }

  private static InitialContext getInitialContext(String url)
       throws NamingException
  {
    Hashtable env = new Hashtable();
    env.put(Context.INITIAL_CONTEXT_FACTORY, JNDI_FACTORY);
    env.put(Context.PROVIDER_URL, url);
   return new InitialContext(env);
 }
```

Listing 2: jmstest.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE jmsml PUBLIC "-//BEA Systems Inc//DTD JMSML Version 1.0
JMSML//EN" "jmsml.dtd">

<jmsml>

  <scenario Name="Dynamic Queue creation" Verify="false">

    <purpose>
To test the dynamic creation of queue
    </purpose>

    <expectedresult>
JMS queue destination should be successfully created with the specified
attributes. In order to verify this, two additional operations are per-
formed as shown below.

1. Take the statistics information of the JMSServer "Warehouse", before
and after the creation of the queue destination, to see if the newly
created queue has been added to the current destination count of the
JMSServer "Warehouse".

2. A sender and receiver will be created to send and receive a text mes-
sage to and from the newly created "orderqueue".

If the above two operations are successful, the test is declared as
PASSED. Otherwise the test is declared as FAILED.
    </expectedresult>

    <operation Class="mbean"
Type="JMSServer"
Name="Warehouse"
Targets="examplesServer"
    >Add</operation>

    <operation Class="mbean"
Type="JMSServer"
Name="Warehouse"
    >List</operation>

    <operation Class="mbean"
Type="JMSServerRuntime"
Name="Warehouse"
    >List</operation>

    <operation Class="mbean" Type="Queue"
Name="orderqueue"
JMSServer="Warehouse"
JNDIName="jms.queue.order"
BytesMaximum="1024000"
BytesThresholdHigh="1000000"
BytesThresholdLow="4096"
MessagesMaximum="100000"
MessagesThresholdHigh="90000"
MessagesThresholdLow="1000"
```

```
PriorityOverride="6"
TimeToLiveOverride="7200000"
DeliveryModeOverride="Persistent"
StoreEnabled="default"
    >Add</operation>

    <operation Class="mbean" Type="Queue"
Name="orderqueue"
    >List</operation>

    <operation Class="mbean" Type="JMSServerRuntime"
Name="Warehouse"
    >List</operation>

    <operation Class="mbean" Type="DestinationRuntime"
Name="orderqueue"
    >List</operation>

    <operation Class="message" Type="Text"
Name="sender"
Factory="weblogic.examples.jms.QueueConnectionFactory"
Queue="jms.queue.order"
Body="Test message to order queue"
    >Send</operation>

    <operation Class="mbean" Type="DestinationRuntime"
Name="orderqueue"
    >List</operation>

    <operation Class="message" Type="Text"
Mode="Synchronous"
Name="receiver"
Factory="weblogic.examples.jms.QueueConnectionFactory"
Queue="jms.queue.order"
    >Receive</operation>

    <operation Class="mbean" Type="DestinationRuntime"
Name="orderqueue"
    >List</operation>

    <operation Class="mbean" Type="Queue"
Name="orderqueue"
    >Remove</operation>

    <operation Class="mbean" Type="JMSServerRuntime"
Name="Warehouse"
    >List</operation>

    <operation Class="mbean" Type="JMSServer"
Name="Warehouse"
    >Remove</operation>

  </scenario>

</jmsml>
```

Listing 3: jmstest.out

```
JMSServer : Warehouse was added to server : examplesServer !
 Name                                  =  Warehouse
 Targets[0]                          =  examplesServer
 BytesMaximum                          =  -1
 BytesThresholdHigh                    =  -1
 BytesThresholdLow             =  -1
 MessagesMaximum               =  -1
 MessagesThresholdHigh         =  -1
 MessagesThresholdLow          =  -1
---------------------------------------------
 Name                                  =  Warehouse
 SessionPoolsCurrentCount      =  0
 SessionPoolsHighCount         =  0
 SessionPoolsTotalCount        =  0
 DestinationsCurrentCount      =  0
 DestinationsHighCount         =  0
 DestinationsTotalCount        =  0
 MessagesCurrentCount          =  0
 MessagesHighCount             =  0
 MessagesPendingCount          =  0
 MessagesReceivedCount         =  0
 BytesCurrentCount             =  0
 BytesHighCount                =  0
 BytesPendingCount             =  0
 BytesReceivedCount            =  0
---------------------------------------------
Queue Destination : orderqueue was added to JMSServer : Warehouse !
 Name                                  =  orderqueue
 JNDIName                      =  jms.queue.order
 JMSServer                     =  Warehouse
 StoreEnabled                  =  default
 Type                                  =  Queue
 BytesMaximum                  =  1024000
 BytesThresholdHigh            =  1000000
 BytesThresholdLow             =  4096
 MessagesMaximum               =  100000
 MessagesThresholdHigh         =  90000
 MessagesThresholdLow          =  1000
 PriorityOverride              =  6
 TimeToLiveOverride            =  7200000
 DeliveryModeOverride          =  Persistent
---------------------------------------------
 Name                                  =  Warehouse
 SessionPoolsCurrentCount      =  0
 SessionPoolsHighCount         =  0
 SessionPoolsTotalCount        =  0
 DestinationsCurrentCount      =  1
 DestinationsHighCount         =  1
 DestinationsTotalCount        =  1
 MessagesCurrentCount          =  0
 MessagesHighCount             =  0
 MessagesPendingCount          =  0
 MessagesReceivedCount         =  0
 BytesCurrentCount             =  0
 BytesHighCount                =  0
 BytesPendingCount             =  0
 BytesReceivedCount            =  0
---------------------------------------------
 Name                                  =  orderqueue
 ConsumersCurrentCount         =  0
 ConsumersHighCount            =  0
 ConsumersTotalCount           =  0
 MessagesCurrentCount          =  0
 MessagesHighCount             =  0
 MessagesPendingCount          =  0
 MessagesReceivedCount         =  0
 BytesCurrentCount             =  0
 BytesHighCount                =  0
 BytesPendingCount             =  0
 BytesReceivedCount            =  0
---------------------------------------------
 Name                                  =  orderqueue
 ConsumersCurrentCount         =  0
 ConsumersHighCount            =  0
 ConsumersTotalCount           =  0
 MessagesCurrentCount          =  1
 MessagesHighCount             =  1
 MessagesPendingCount          =  0
 MessagesReceivedCount         =  1
 BytesCurrentCount             =  27
 BytesHighCount                =  27
 BytesPendingCount             =  0
 BytesReceivedCount            =  27
---------------------------------------------
@receiveMessage() : Message Received [0] : Test message to order queue
 Name                                  =  orderqueue
 ConsumersCurrentCount         =  1
 ConsumersHighCount            =  1
 ConsumersTotalCount           =  1
 MessagesCurrentCount          =  0
 MessagesHighCount             =  1
 MessagesPendingCount          =  0
 MessagesReceivedCount         =  1
 BytesCurrentCount             =  0
 BytesHighCount                =  27
 BytesPendingCount             =  0
 BytesReceivedCount            =  27
---------------------------------------------
Queue : orderqueue was removed from JMSServer : Warehouse !
---------------------------------------------
 Name                                  =  Warehouse
 SessionPoolsCurrentCount      =  0
 SessionPoolsHighCount         =  0
 SessionPoolsTotalCount        =  0
 DestinationsCurrentCount      =  1
 DestinationsHighCount         =  1
 DestinationsTotalCount        =  1
 MessagesCurrentCount          =  0
 MessagesHighCount             =  1
 MessagesPendingCount          =  0
 MessagesReceivedCount         =  1
 BytesCurrentCount             =  0
 BytesHighCount                =  27
 BytesPendingCount             =  0
 BytesReceivedCount            =  27
---------------------------------------------
JMSServer : Warehouse was removed from server : examplesServer !
---------------------------------------------
```

Attention All BEA Developers: Stop Fearing Mainframe Integration

DEVELOPERS AND MAINFRAME PERSONNEL CAN WORK WELL TOGETHER

BY **DOC MILLS**

There are many reasons why organizations fear mainframe integration. Proprietary interfaces, radically different processing environments, lack of support for standard development APIs, and the fact that the people who created the applications have since “moved on” are the most common factors identified when an organization postpones a mainframe integration project.

These are all reasonable issues, but in the same breath, these issues must be resolved for organizations to fully realize and maximize their investment in mainframe data.

There are an equal number of compelling reasons why organizations should deploy mainframe resources as part of WebLogic application development and integration projects. In this article, I'll review the current state of the industry and discuss the solutions needed to create a comfort zone that allows application developers to access mainframe resources.

Current State of Affairs

The leading reasons for using mainframe resources in conjunction with WebLogic application development and integration projects include leveraging existing logic and data, increasing the value of current investments, and focusing development effort on solving new business problems. For users of the BEA WebLogic Platform who are building high-value applications that depend on IBM mainframe data and applications, getting this integration right means avoiding the headaches associated with a deluge of extra costs and specialized training.

There's a line of demarcation that divides WebLogic development organizations and mainframe systems management organizations. You can't see it, but you know it's there. From a productivity perspective, it's the boundary between comfort zones. WebLogic developers are resistant to learning about the mainframe, and the mainframe “glass house” personnel are focused on maintaining the security and stability of their tightly controlled production environment, preferring to avoid interaction with the chaotic world of open systems applications.

From a technological perspective, “the line” is an effectiveness threshold. Each computing environment has strengths and weaknesses. When a solution pushes the capabilities of a platform too far, problems are inevitable.

Fear and frustration often surround the idea of mainframe integration because most solutions force someone to cross “the line.” The results are scalability and failover problems, limited feature set support, additional administrative headcount requirements, ongoing training, and maintenance rollout problems. For the organization, the results are development delays, a backlog of support calls, higher Total Cost of Ownership, and, more important, dissatisfied internal and external clients.

First Step: Identify the Problem

The first step in communication resolution is always the hardest. We must take the time to identify the problem before we can begin to fix it.

In finding a solution to mainframe integration, the logical question is, “How can I move forward without crossing ‘the line’?” As with any communication problem, one idea is to use an arbitrator to ensure that individual strengths are communicated and a common solution is achieved. This principle can be applied to mainframe integration as well.

Technologists are not the best long-term strategists. The point is that we need to employ cross-communication solutions that not only solve present problems but also provide the foundation to address future obstacles.

Outline an Ideal Solution and Stay Within Your Comfort Zone

The next step in jumping the communication hurdle between WebLogic developers and mainframe administrators is to outline an ideal solu-

AUTHOR BIO

Doc Mills is currently the director of product marketing for NEON Systems, the leader in enterprise-class mainframe adapters that reduce integration complexity. He has more than 18 years of experience with mainframe networking and integration.

CONTACT...

doc.mills@neonsys.com

WebServices
.NET J2EE XML JOURNAL

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

LEARN WEB SERVICES. GET A NEW JOB !

SUBSCRIBE TODAY TO THE WORLD'S LEADING WEB SERVICES RESOURCE

Get Up to Speed with the Fourth Wave in Software Development

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

SYS-CON MEDIA

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

Once you're in it...

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

reprint it...

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

SYS-CON MEDIA

RePrints

THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

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

Helping you enable intercompany collaboration on a global scale

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

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

Go Online & Subscribe Today!

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

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

SYS-CON MEDIA

tion. This is an important step and requires the most effort. We can all identify the problem, and once we have the tools in place, we can start fixing it. The important step in between is ensuring that each group is comfortable with the proposed solution, and once that solution is in place, is committed to “cross-the-line” communication.

One idea is to place a communications component on each side of “the line.” In this manner, the communications component, or arbitration component, presents each organization with an interface that looks and acts in a familiar manner. For WebLogic developers, this means working with standard development tools where mainframe resources behave like distributed relational databases. For mainframe administrators, the arbitration component provides the enterprise-class monitoring, management, and control facilities required to maintain system availability and ensure minimal utilization of CPU.

Here are several points to keep in mind when building an ideal solution. Remember to think long term. Put together a roadmap that evolves with changing business and technology needs. Find a solution that supports application integration and data integration. If your organization requires access to programs running under IMS/TM or CICS and access to the underlying data stored in DB2, IMS/DB, VSAM, and other databases, search for a solution that provides the WebLogic platform with standard API access to the most important mainframe transaction managers and databases. By doing this, you’ll fix today’s problems and have a plan to fix issues lurking around the corner, saving time and money along the way.

Communication problems arise when we are forced into uncomfortable situations. The same is true with mainframe integration. To solve this problem we must enable a “cross-the-line” solution to communicate on our behalf. This solution must be transparent to the WebLogic platform, exploiting native OS features for performance optimization and addressing the translation and connectivity issues that can be handled from the distributed platform. To developers, mainframe databases must appear as distributed systems’ relational databases, and mainframe programs must look and behave like stored procedures.

During the production phase, the solutions must support connection pooling and two-phase commit (2PC) transactions through extensions compatible with the Java Transaction Service (JTS) and the Java Transaction API (JTA). The solutions should also gather diagnostic and performance data for debugging and troubleshooting purposes, ideally through a choice of J2CA, JDBC, and ODBC adapters, and an agent that gathers information useful in troubleshooting.

Part of the “fear factor” that mainframe integration has evoked over the years has been self-inflicted by component architectures that have been less than adequate and that caused problems and confusion from development through deployment. This is directly due to taking individual users out of their comfort zone. In many instances these substandard architectures have a direct impact on overall performance.

Screen scrapers usually fall into this category but can also be found in mainframe integration paradigms that require multi-tiered server architectures. While there can be some instant gratification with screen scraping, performance will always be a problem and administrators can expect to be continually adding additional network servers and the complexities associated as throughput demands increase. With screen scraping, XA two-phase commit and direct data access go by the wayside, limiting functionality. Lastly, this paradigm requires someone on the Web server side to understand the application they are trying to access from the host. Other issues soon surface, like trying to “record” mainframe terminal sessions and the maintenance associated when mainframe applications change. These tasks take individuals out of their comfort zone.

Likewise, there are problems with multi-tiered mainframe integration architectures. These architectures, usually conceived off the mainframe, then eventually migrated to it, have one or more gateways or servers associated with the open Web server side. Some in this category even have one or more mainframe footprint requirements. While these may not be dubbed screen scrapers and may not require someone with mainframe expertise to manage or maintain on the Web server side, they do present other problems. The problem associated with this architecture is one of performance coupled with the fact that there is no special attention paid to any one operating system. Why run on the mainframe if you don’t take advantage of any of the benefits associated with mainframe architectures or subsystems This paradigm also has the potential of taking the mainframe personnel out of their comfort zone by having them install, configure, and maintain a component designed and implemented on a Unix/NT platform, within a mainframe address space. In a nutshell, the more moving parts, the more complex the architecture, the greater the likelihood that it will have a point of failure. Worse yet is trying to track down where the failure occurred.

The most effective architecture will allow users and administrators to keep within their comfort zones and satisfy users on both sides of the imaginary “line.” To the integrator or Web application developer, this provides for consistent standards such as JDBC, J2CA, and JCA in a thin-client paradigm. These personnel do not want to get bogged down with terminologies that are foreign to them. They also want to be assured that the architecture they choose does not have a major performance impact on them, or the rest of their environment, while maintaining WebLogic security, clustering, loadbalancing, and failover. Likewise, the mainframe systems programmer or DBA demands security and a product that has superior performance and a resource-prudent architecture that can meet the needs of thousands of simultaneous user requests, as well as allowing them the ability to monitor and control their own environment. Products of this nature can only have a mainframe footprint that makes use of the mainframe subsystems and subtasking that have made the mainframe the cornerstone of reliability, availability, and security (RAS).

Remember the main goal: find a solution that allows developers with no knowledge of the mainframe or mainframe access technology to build effective applications across all components of the WebLogic Platform, allowing for the use of one tech-

“In finding a solution to mainframe integration, the logical question is, ‘How can I move forward without crossing ‘the line’?’”

A"CROSS-THE-LINE COMMUNICATION" SOLUTION: WEBLOGIC AND MAINFRAME WEB SERVICES

Organizations that want to use Web services within the WebLogic Platform to make Simple Object Access Protocol (SOAP) requests to the mainframe can deploy a J2CA enterprise-class adapter solution that can be configured to accept inbound SOAP requests from TCP/IP at the mainframe, and map the requests to a particular CICS or IMS/TM application. The SOAP request can be built in WebLogic Workshop based on WSDL stored in the adapter solution's metadata repository. This capability allows organizations to begin creating and exploiting Web services that access the mainframe in a manner that maintains the value of existing investments.

nology across multiple projects. This isn't as easy as it sounds, but if you keep this objective in mind, you'll solve current and future "cross-the-line" problems without a detrimental effect on the business.

Don't Forget Their Needs

Let's take a quick look at what is needed on the mainframe side of the "line." The ideal solution fully understands application development and mainframe systems, leaving professionals on each side of the line to focus on the task at hand.

A true "cross-the-line" solution simplifies monitoring, management, and control of composite applications for mainframe systems administrators. While some organizations would prefer a solution that did not include a solution running on the mainframe, the reality is that this solution is absolutely required in order to ensure high performance, scalability to meet evolving business and technological needs, and a manageable operational environment. Ineffective solutions are developed and maintained independently, of each other and scattered throughout multiple mainframe subsystems, and do not provide a consistent or comprehensive integration infrastructure for the mainframe.

There is a reason why mainframes have been the de facto standard for security, availability, and reliability for more than 30 years. You need to deploy a solution that extends these features to composite applications, providing maximum reliability and minimum resource utilization without compromising its ability to support thousands of users and thousands of transactions per second. Most important, the solution must mask the complexity of the mainframe.

Once in a While, We Must Step Over the Line

There is one aspect of composite applications where it is absolutely required that developers and mainframe systems personnel step over "the line" – application debugging and problem resolution. In this situation, developers and systems administrators must work together to find the cause of application or performance problems. The process normally entails gathering and reviewing trace logs from the application, any midtier servers, the mainframe integration technology, and the mainframe resource itself.

Unfortunately, if there is a weak spot in many component architectures, it surfaces in the area of problem determination and diagnostics. The last thing an application developer wants to do is to call a mainframe systems programmer and ask for a GTF trace, to piece together with the information they have accumulated from the open Web side. Likewise, the systems programmer doesn't want to spend a lot of time explaining a trace that the application developer doesn't understand in the first place.

The ideal solution will provide a communication interface that shows application calls and mainframe responses, allowing administrators to change operational parameters, and providing "drill-down" capability that displays response time and other critical information that helps administrators maintain service levels and detect problems. In the end, simple problems are corrected automatically, and complex problems can be located and corrected with less effort and disruption.

This type of solution is equally important to application developers. Look for a solution that will enable trace and diagnostic aggregation between mainframe and open-system components, eliminating the need for the dreaded call to the mainframe Systems Programmer to help piece information together. A telltale sign of a good component architecture will show little to no performance impact with full trace diagnostics enabled. This eliminates the need for problem replication, speeds up the diagnostic process, and allows both developers and systems administrators to remain productive and work within their comfort zones.

This type of solution is equally important to application developers. Diagnostic data presents a view of events from the WebLogic Platform to the mainframe and back, allowing developers to troubleshoot and tune applications with little or no assistance from mainframe systems administrators. This significantly reduces the time required to find and fix problems and allows both developers and systems administrators to remain productive and work within their comfort zones.

Cross-the-Line Communication Removes the Fear

True cross-the-line communication allows us to recognize the challenges that face our mainframe counterparts without leaving our comfort zone. There are many points to keep in mind when searching for a solution. After identifying the problem, spend time outlining an ideal solution that will solve immediate problems and evolve with the organization to meet future challenges.

Next, it's important to evaluate available solutions with a critical eye, knowing that the best solution will provide an agnostic platform for application developers and mainframe administrators to stay within their comfort zone.

By not forcing any organization to cross "the line," developers and systems personnel work well within their comfort zones, computing systems work effectively within standard application implementations, and the overall organization gains increased productivity and business agility.

Convergence: The

A BIG STEP FORWARD

BEA eWorld 2003, BEA's annual technology conference, was held on March 2–5 at the Gaylord Palms Resort and Convention Center in Orlando, Florida – a spectacular all-in-one facility that boasted lush gardens and great meeting spaces.

BEA founder, chairman, and CEO Alfred Chuang kicked off the show on Monday morning in a cloud of smoke and laser lights addressing the more than 2,000 delegates on the conference theme – convergence: bringing together application development and application integration into one unified framework. He then went on to introduce BEA WebLogic Platform 8.1, which standardizes the application infrastructure platform so businesses can leverage their legacy systems and build the applications they need.

Chuang was followed by Carly Fiorina, chairman and CEO of HP, who talked about HP's partnership with BEA, saying it allowed the two companies to deliver solutions that built on each of their strengths. She also announced that HP is working with BEA to create a dedicated Web services management and deployment practice for J2EE solutions within HP's Services group. She talked about both HP and BEA as leaders in open standards, a commitment that separates them from many of their competitors.

Following the opening keynotes, the conference delegates attended a wide array of technical sessions on all of the new features of WebLogic Platform 8.1, including classes on process integration; Web services; integrated portals; integrated development strategies; performance, scalability, and security; and standards (as Carly Fiorina said, BEA is a leader in its commitment to open standards) – all of the issues addressed in the new platform.

On the busy exhibit floor, show sponsors Intel, HP, Documentum, Borland, Accenture, Hyperion, Sun Microsystems, Siebel, Hyperion, and close to 100 more companes were available to talk about what they can provide to BEA users and those looking to implement applications on the leading app server platform.

WebLogic Developer's Journal, a media sponsor, not only had a booth featuring the March issue – which included all of the details on the latest release as well as commentaries and technical pieces from some of BEA's partners – but also brought along the popular SYS-CON radio to provide interviews with the leaders of some of BEA's partners as well as with company executives. (The interviews are available online at www.sys-con.com/weblogic.)

On Tuesday morning, Olivier Helleboid, president of BEA's Products Organization, kicked off the morning's

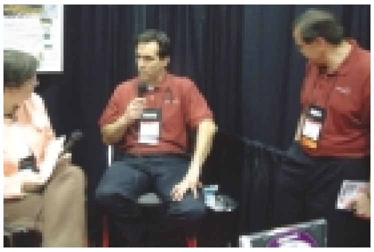
AltaWorks: Michael Courtemanche, Chief Technology Officer (left), and Campbell Stras, VP of Marketing (right)

BEA Systems: Byron Sebastian, Vice President and General Manager, WebLogic Workshop and Portal

Cisco Systems: John Yen, Senior. Product Marketing Manager, Content Networking

Cyanea: Bob Lee, VP, Technical Marketing

Dirig Software: Jeremy Davis, President/CEO (left), and Dave Wilby, Vice President of Product Management (right)


PANACYA: Brian Murphy, Vice President North American Sales

Quest Software (Sitraka): Ed Lycklama, Chief Technology Officer & Co-Founder of Sitraka (now a part of Quest)

events with a keynote address on BEA WebLogic Platform 8.1. In just one year, BEA has delivered an integrated application infrastructure platform that provides a common runtime for J2EE applications, integration initiatives, and portal projects. In the new release, WebLogic Workshop 8.1 plays a leading role. WebLogic Workshop 8.1 allows different kinds of developers, including architects, database developers, J2EE developers, portal developers, and to work in the same environment. New applications will run on improved versions of all of BEA's server-based products, including WebLogic Server, Tuxedo, WebLogic Integration, Liquid Data for WebLogic, and WebLogic Portal.

Helleboid was followed by John Davies, vice president of Intel Architecture Group and director of its Solutions Market Development Group. Davies also talked about the conference theme – Convergence – and said Intel is looking at hardware convergence – the meeting of computing and communications. There were also several technical keynotes. On Monday Adam Bosworth, BEA chief architect and senior VP, Advanced Development, talked about BEA WebLogic Workshop 8.1, and its creation of an environment where the system programmers, application developers, and business users can work together, sharing the same code. Tuesday afternoon, BEA CTO Scott Dietzen looked at Web 2.0, and talked about some of the best practices that will get the most out of Web services–based applications. Scott returned to the podium the next day to build on what he said on Tuesday with an overview of some of the standards that he believes will enable his idea of Web 2.0.

Once again, the exhibit floor was busy with delegates looking at everything to meet their development needs, especially for the new release. In talking with some of the exhibitors, ***WLDJ*** found that they thought the show was a total success and were very happy with the contacts they made. They all plan to be back again next year – BEA eWorld 2004 will be held in San Francisco May 25–28.

The Little Extras

This year, BEA offered a wireless hub to its attendees. A beanbag chair – furnished area was provided near the conference sessions so delegates could check their e-mail and surf the Web without plugging in. And a comfortable seating area was also available on the exhibit floor for the same purpose.

Three BEA leaders – CEO Alfred Chuang, CTO Scott Dietzen, and Chief Architect Adam Bosworth – were immortalized as bobblehead dolls that were given out to all of the attendees.

On Tuesday night, eWorld delegates went to the Convergence Party at Universal Studios. They rode on one of the world's fastest roller coasters, danced with superheroes, and filled the video arcade at Marvel Super Hero Island.

In all, another successful conference, packed with information, education, and products. ***WLDJ*** looks forward to seeing you all at BEA eWorld 2004!

BEA: Scott Dietzen Chief Technology Officer offered a technical Keynote on the Web 2.0

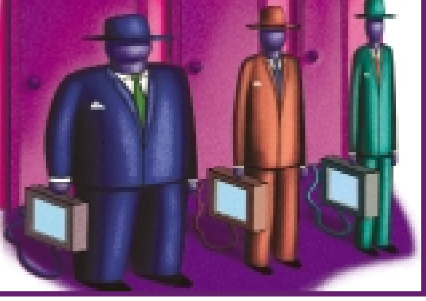
Gail Schultz, executive editor of *WLDJ*, spoke to eWorld exhibitors for SYS-CON Radio

Wily Technology: Chris Farrell, Director of Product Marketing

Precise Software: Thomas B. Mulvehill, Product Line Manager Web-Based Technologies

NuWave Solutions: Howard Block, Principal

BEA Systems: Scott Fallon, Vice President, Developer Relations



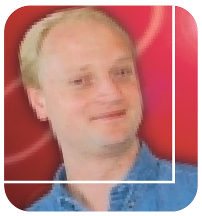
Transactions – Just Another Tinker Toy?

MOVING COMPLEXITY INTO OBSCURITY

BY PETER HOLDITCH

Over the past 16 months – has it really been that long?! – I have attempted to climb the peaks of how to design applications that use transactions, and dived into the depths of the earth, looking at obscure knowledge such as how clients can demarcate transactions, and grubby details of interfacing WebLogic's transaction manager with external resources like MQ Series and, much to my surprise, you have come along on this odyssey with me. Who'd have thought that there were so many people interested in these minutiae?

Well, to be honest, I would for one. Being an engineer by training, I like nothing more than tinkering with these kinds of mechanisms. It's interesting, challenging stuff. Some people play chess. I don't know the rules so I need an alternative pointless endeavor! But that is only half of engineering. The world wouldn't contain any engineers at all if they never produced anything – if we all spent our lives tinkering in our sheds, instead of building software systems, bridges, roads, cars, and so on, who would care? We'd be begging on the street corners, transaction manager in one hand, with a mangy dog holding a JDK in its mouth, and no more than an old copy of Kernighan and Ritchie to sleep under at night. That's not the case. We do build things too.



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 things we build come in two categories. On the one hand, there is the tedious, repetitive "yet another" forms application that gathers some information from a user, updates some databases, and repeats until bored (or accidentally powered down by the cleaner). On the other hand, sometimes our employers need very technical stuff done, because unless their business is run on some pretty revolutionary systems, with mind-numbing response times and so on, they won't be able to compete and we'll all be out of a job, fighting over the K&R, and who gets first bite of the dog. For the second category of engineering problem, burning the candle at both ends reading transaction articles is time well spent – with a deep and thorough understanding of the subsystems that make up the underlying infrastructure (and a healthy dash of experience and some luck) you can build something truly amazing – the Sydney Opera House of applications.

Who Gets First Bite of the Dog?

However, the Sydney Opera House wouldn't make a great affordable housing scheme. Here builders with more modest aspirations can happily erect a housing complex from prefabricated parts just like they did last week, and will again next week, and make a family or several very happy. What is needed here isn't the pinnacle of creative expression, it's a good job well constructed from components and templates with the minimum of cost and fuss.

So what on earth is he rambling about now, I hear you sigh, has he finally lost it?

Well, not quite. To return from the land of metaphor for a moment, sensible use of transactions will be necessary whether you're building an order entry forms app, or a world-beating innovation. It's just that in the former case, there are many widely applicable and well-understood design patterns for their use (you know, the stuff I wrote about before) that can be applied in the knowledge that they will "just work" because the problem is so similar to so many others. Under these conditions, it should be possible to forget almost entirely about transactions and get on with implementing business logic. After all, how many construction engineers worry on a day-to-day basis about the laws of physics? Transactions really should be fundamental laws–type stuff…

Now, imagine a world where you can graphically describe your business process, add the few lines of code that actually do your processing at the right points, and press "run." Imagine also that you can design (or even generate) the forms front end at the same time. Sounds like second-generation, client/server desktop development tools. Imagine that you can deploy the results to large, and variable, user populations and it all still works… That sounds new! It also sounds like the new (8.1) version of WebLogic Platform. The Workshop tool allows you to do exactly that. But make no mistake, this goes much deeper than a tool. Anybody can host a load of windows in a single environment and produce a "developer tool portal," but that's not where the productivity comes from.

Remember, the productivity came from the assembly patterns and components, not from having a "universal hammer" that screws while it nails while it drills. The depth, and the value, is in the runtime framework that underpins the development environment. It applies sensible patterns and standards across the board, allowing you to treat low-level considerations like transactions as laws-of-physics rather than matters of pressing concern.

It Screws While It Nails, While It Drills!

So how does this relate to transactions directly? Well, in the process designer you draw your business process. If it's a long-running process, the framework will start a transaction at the start of the process and commit it whenever the process comes to a pause, to ensure the state is preserved. Probably what you'd want, right? For a process that doesn't pause it will start a transaction, do all the work, and simply commit it. Probably what you'd want too, right? But if the probabilities fail you, you can rubber band the transactions over your process steps, and tell the framework where they should start and end based on the physical realities of your system, still without a line of code explicitly referencing transactions. Oh, and by the way, when the framework assumptions don't apply to your particular use case you can drop back down to the J2EE APIs and the good old application server knowing that all the code will still end up deploying into the same environment.

Back to the Future

I remember my first column (***WLDJ***, Vol. 1, issue 1) talked about how transactions were assumed to be complex because all the two-tier developers never had to think about them, they just magically happened. And for the majority of applications, that's the way it should be. More strength to BEA's platform elbow in pushing them back into obscurity where they belong!

This month, I'll look at benchmarking and tuning your applications, and how to make your Java runtime more manageable. And, I offer some advice on how your developers can keep their focus on development work.

Benchmarking, Tuning, and Manageability

AND GETTING BACK TO THE HEART OF YOUR JOB

BY LEWIS CIRNE

AUTHOR BIO

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

CONTACT...

asklew@sys-con.com

Q. CAN YOU RECOMMEND A PROCESS FOR BENCHMARKING AND TUNING APPLICATIONS?

A. To establish baselines on a production application, you need to determine its performance characteristics prior to deployment. The end result of this process is twofold. First, it allows you to tune the application to defined business objectives and verify it against a set of load tests that closely match expected production conditions. Second, it helps you to identify the application components that are critical to performance so that they can be monitored and tracked in production. The next step is critical to any capacity planning efforts as the expected load on the application grows.

I suggest that you take the following steps:

1. Create a set of test scripts that mimic user interactions.
2. Identify a performance characteristic to monitor (e.g., response time, CPU time, I/O activity).
3. Determine the characteristic's target performance level.
4. Using the test scripts, generate load on the application to establish a baseline for the performance characteristic under investigation.
5. Identify components of the application driving the performance characteristic and apply methods, one by one, to improve them.
6. After each applied performance improvement, rerun the tests to determine the level of improvement, if any, from the baseline.
7. Repeat steps 5 and 6 until the target performance level is achieved.

The importance of steps 3 and 4 cannot be overstated. Without a performance baseline, investigators cannot determine the effectiveness of any improvements nor can they judge if the performance goals can reasonably be attained. When both the generated load and the test environment sufficiently mirror production circumstances, I have found this process to be highly successful in helping ensure high application performance over the long run.

Q. HOW WILL THE JAVA RUNTIME BECOME MORE MANAGEABLE OVER TIME?

A. Historically, Java has done a great job of servicing the needs of the developer by rapidly rolling out new APIs and services developers need to build functionality and get the application functioning rapidly. However, when it comes to manageability the JVM is one area where there is room for improvement. For example, the Java runtime does not provide APIs for notification about garbage collection or for measuring thread performance.

What organizations need are APIs for performance and availability that specifically target production environments and incur minimal overhead. Those APIs are currently being specified – there are some new JSRs, such as JSR 174, that focus on exactly these issues. While the standards bodies address performance and availability, some JVM vendors have begun efforts to make their JVMs more manageable because they understand that there's an immediate need for a solution like this at the JVM level itself. (JRockit, for example, offers extra features for manageability.)

JMX offers some performance and availability management functionality, and organizations should consider coding APIs that allow them to utilize it. The primary strength of JMX, however, is in making the application servers themselves more manageable, and that's the focus of much of the current JMX research and implementation. Over time, manageability will become more and more a part of the standard Java runtime. This will be enabled by the emergence of new JSRs that make the JVM more manageable with low overhead in a production environment and that focus on performance, availability, and resource utilization.

Q. I'M A DEVELOPER AND I KEEP GETTING PULLED INTO MEETINGS TO SOLVE PRODUCTION APPLICATION PERFORMANCE AND AVAILABILITY PROBLEMS REGARDLESS OF THEIR CAUSE. WHAT CAN I DO TO GET OUT OF THIS AND GET BACK TO DEVELOPING SOFTWARE?

A. This is a problem that many developers have asked me about. Too often, software developers get pulled in to solve production problems without much information about what the cause of the problem is. As a result, they spend needless time looking through thread dumps and log files trying to find a needle in the haystack, rather than spending their time developing software.

There is a need for a specific role within the company called a Level II application support manager. This person serves as a complement to the day-to-day operations people who handle typical management duties such as making sure the hardware boxes are rebooted on time or that DNS servers are available. When there's a problem with an application, the primary role of the application support manager is to perform triage. He or she should know the right person to call to get the problem fixed and be able to provide that person with information about why they were called in and why their area of expertise is needed. For example, they should be prepared to offer some trending information, some historical performance information, or some real-time performance information to whomever they call so that person isn't thrown into complete darkness.

If you have an application support manager to monitor the application and who can perform triage quickly and accurately when there's a problem, then the developers will be taking fewer calls, and the ones they do take will legitimately require their expertise. This will result in serious productivity gains for the development organization as well as serious gains in application availability on the operations side.

* * *

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

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altaworks	www.altaworks.com/wldj	888-877-7111	52
BEA Systems	http://dev2dev.bea.com/useworkshop	800-817-4BEA	51
BMC Software	www.bmc.com	800-841-2031	15
Borland	www.agile.borland.com/wd1	831-431-1000	11
Hewlett-Packard	http://devresource.hp.com/wldj.html	800-752-0900	27
Informatica	www.informatica.com	800-653-3871	23
JDJ Workshop	www.sys-con.com/education	201-802-3058	29
Linux Business & Technology	www.sys-con.com	888-303-5282	35
PANACYA	www.panacya.com	877-726-2292	9
PowerBuilder Developer's Journal	www.sys-con.com/powerbuilder	888-303-5282	43
Quest Software	http://java.quest.com/jcsv/wldj	800-663-4723	21
Quest Software	http://java.quest.com/performance/wldj	800-663-4723	3
ReportingEngines	www.reportingengines.com/download/wldj1.jsp	888-884-8665	2
SYS-CON E-Newlsetters	www.sys-con.com	201-802-3020	47
SYS-CON Media	www.sys-con.com	888-303-5282	37, 47
SYS-CON Publications	www.sys-con.com	201-802-3021	33
SYS-CON Reprints	www.sys-con.com	201-802-3026	41
Web Services Journal	www.sys-con.com	888-303-5282	41
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	41
Wily Technology	www.wilytech.com	888-GET-WILY	4
Wysdom	www.wysdom.com	877-499-7366	17

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

LOOK WHAT'S COMING NEXT MONTH

Diagnosing Application-Database Performance Bottlenecks
Properly diagnosing J2EE performance problems requires visibility into three components: WebLogic resource utilization and configuration, the application architecture, and the database query execution.

The Race to Create Web Services Business Process Standards
The number of Web service business process (BP) specifications trying to make their way to standards status makes it difficult to tell who is doing what. We'll make sense out of the morass by classifying Web service BP specifications into four categories.

Enhancing Application Manageability
The absence of manageability can cause serious problems. Application software can become an unmanageable entity once it is deployed because the operations staff does not understand it. Lack of attention to manageability raises the cost of supporting and maintaining the software product significantly. This article offers choices to developers on solving these problems.

JMS Messaging and WebLogic 7.0
Creating a helper class for sending JMS messages generically with WebLogic is beneficial for any asynchronous messaging effort, and provides a basis for future JMS development. The advantages of creating a generic access path for JMS are highlighted, and a method for doing so is provided.

J2EE and Struts
What can be accomplished by using Struts? As a start, you can standardize navigation, validation and display, and make your applications easy to use, maintain, and extend.

Plus, review of WebLogic Workshop 8.1

BEA WebLogic DEVELOPER'S JOURNAL

News & Developments

BEA Announces General Availability of WebLogic Server 8.1

(San Jose, CA) – BEA Systems, Inc., the world's leading application infrastructure software company, has announced the general availability of BEA WebLogic Server 8.1 and BEA WebLogic JRockit 8.1, the cornerstones of BEA's converged application platform suite.

As the base of the BEA WebLogic Enterprise Platform, BEA WebLogic Server 8.1, an enterprise application server, is designed for building service-oriented Java applications that help solve the biggest challenge facing enterprises today – business integration. With this release, BEA introduces new functionality designed to shorten development cycles, increase operational efficiency, automate business processes, and deliver an adaptive foundation that can change at the speed of business – all while reducing overall costs. www.bea.com

Confluent CORE 3.0 Platform Offers Service-Oriented Architecture Now

(Sunnyvale, CA) – Confluent Software Inc., has announced Confluent CORE 3.0, the latest version of its flagship Web services monitoring and active, policy-driven management platform. This version provides enterprise-class security management, a flexible deployment model, management for Java and .NET services right out of the box, and integration with leading systems management products.

CORE 3.0 is the first deployed Web services management platform that natively supports both Java and .NET applications. www.confluentsoftware.com

WebLogic Enterprise Platform Foundation For NTT Service

(San Jose, CA) – BEA Systems, Inc., has announced that Nippon Telegraph and Telephone, a telecommunications service provider, has selected the BEA WebLogic Enterprise Platform as the foundation for its L-Mode service.

The service, available throughout Japan, connects individuals to the Internet through traditional telephones and fax machines equipped with small L-Mode digital displays. www.ntt.co.jp

Panscopic Enhances Analytic Reporting with WebLogic Platform

(San Francisco) – Panscopic, a provider of Web-based analytic reporting solutions, has announced the availability of Panscopic's native J2EE analytic reporting solution on the BEA WebLogic Enterprise Platform. Its integration with BEA WebLogic can extend the current functionality of the solutions it is used with, providing overall enhancement of applications. www.panscopic.com

UICI Builds Interactive Enterprise Portal on BEA WebLogic

(San Jose, CA) – UICI, a provider of insurance and financial services for individuals and families, has built a collaborative employee portal on BEA WebLogic Portal, improving company-wide productivity and communication while reducing operating costs.

With its employees working at locations throughout the country, UICI deployed an enterprise portal utilizing BEA WebLogic Portal as a simplified means of maintaining strong intra-company communication, eliminating redundant processes reducing operating costs, and maximizing the productivity of employees and agents whose work requires interaction with people based in different offices. www.bea.com, www.uici.com

EOC Selects the WebLogic Platform for Internet Trade-Clearing Utility

(San Jose, CA) – EnergyClear Operations Company, LLC (EOC) has built the first standards-based over-the-counter (OTC) Internet trade-clearing utility on the BEA WebLogic Enterprise Platform and the onExchange Extensible Clearing System (ECS).

EOC provides and manages the facilities and technology necessary for the operation of clearinghouses for OTC commodity markets. They cited the extreme reliability, scalability, and the unmatched ease of integration that the BEA WebLogic Enterprise Platform provides as factors that make it the ideal foundation for the clearing application. www.energyclear.com

Cyclone Commerce and BEA Offer First Application Platform Suite

(San Jose, CA and Scottsdale, AZ) – BEA Systems, Inc., and Cyclone Commerce, Inc., producer of collaboration management solutions, have announced an agreement whereby BEA WebLogic Platform 7.0 will become the industry's first application infrastructure software to support Cyclone Central, a trading partner collaboration management application. In addition,

Cyclone Commerce announced that it intends to support BEA WebLogic Platform 8.1 and BEA WebLogic Workshop 8.1, and intends to build prepackaged resource interfaces, called controls, for BEA WebLogic Workshop 8.1, BEA's unified development environment.

Cyclone Central is an open, standards-based system that rapidly ramps, manages, monitors, and defines community policies for an entire network of hubs and endpoints. www.cyclonecommerce.com

BEA Systems

http://dev2dev.bea.com/useworkshop

Altaworks

www.altaworks.com/wldj