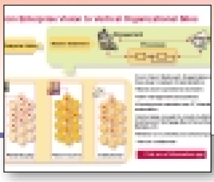
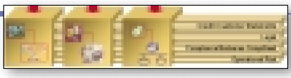
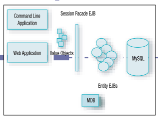
WHO WILL WIN ON THE DESKTOP? PG.4

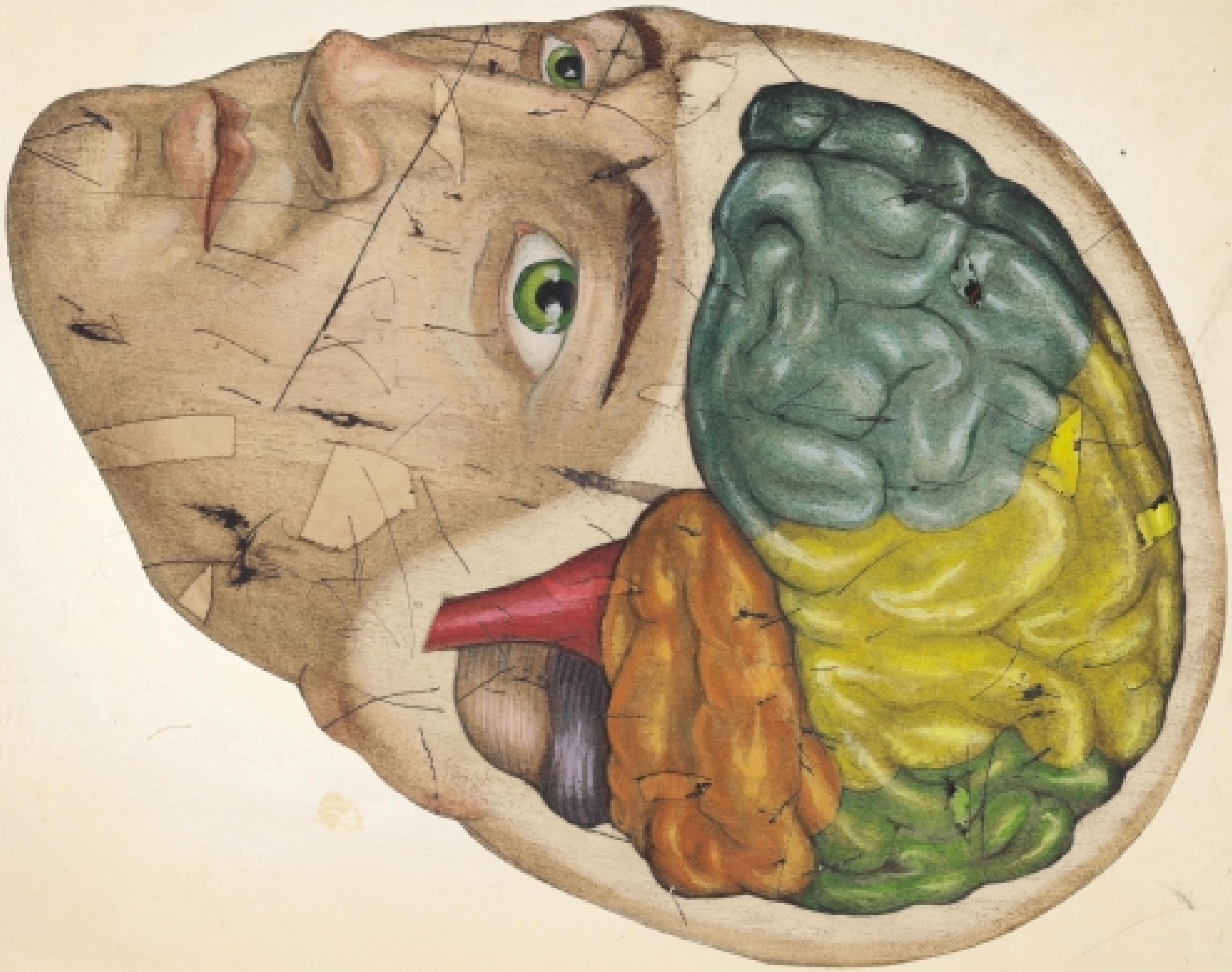
BEA WebLogic DEVELOPER'S JOURNAL

APRIL 2004 - Volume:3 Issue:4

RETAILERS PLEASE DISPLAY UNTIL MAY 31, 2004
$9.99US $9.99CAN
04
0 74470 03424 7

An Architectural Blueprint



DO YOU HAVE A BRAIN?

CEREBRUM: CONTROLS THOUGHT.

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

MEDULLA OBLONGATA: CONTROLS INVOLUNTARY FUNCTIONS.

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

BROCA'S AREA: CONTROLS LANGUAGE.

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

PITUITARY GLAND: CONTROLS HORMONES.

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

YOU HAVE A BRAIN.

So why spend so much time on mindless coding?

Download Weblogic Workshop Now.

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

bea®

The Second Battle for the Desktop

BY JOE MITCHKO

We often like to assume that most corporate IT organizations have kept somewhat up-to-date with all of the various technological innovations over the years, and have done so in an incremental manner. However, the reality of the situation is quite different. You may (or may not) be surprised by how many IT organizations do not necessarily ride the "bleeding edge" wave for one reason or another.

It's not that these IT shops have not paid attention to what is available to them. Now poised to leapfrog over a generation or two of architectural change, these companies are busy looking to overhaul their IT infrastructure and are in the midst of evaluating new architectural approaches. However, which architectural platform and standard should they choose?

For many, it comes down to a choice between two leading contenders – J2EE-based application server companies such as BEA, with their open-standards approach to enterprise computing; and the not-so-new kid on the block, Microsoft, with their .NET-powered line of server and development tools.

Prior to the dawn of Web services, it was a whole lot easier to differentiate between Java and Microsoft. At that time many considered the Java language and platform to be a solution to many problems, including software portability. The write once, distribute everywhere mantra led us to believe that it was the cure for all of our ills. Moreover, with the advent of browser-based applications, made possible by J2EE-based Web servers, our software distribution problems were finally over too. For once, it looked like there was an overall approach to solving a number of the problems that typically plague the application development life cycle.

Of course, technology never stands still and Microsoft did not sit on its hands during this period and watch J2EE take over the corporate enterprise standard. The result of their work culminated in what we know today as the .NET platform – a technology solution that to some mimics and improves on J2EE in a number of ways. All of a sudden, the choice is much more complex, and the differences between J2EE and .NET are not as obvious as before.

Now, we have to aim higher on the technology stack to see where the real differences exist. Since most J2EE platform vendors and Microsoft now provide a business process management engine (BEA WebLogic Integrator versus BizTalk for example), that doesn't work as a differentiator either. We need to look elsewhere. We need to look at how well each platform can integrate business process management (BPM) with human workflow activity.

In the not so distant future, corporations will be looking for new ways to merge automation and manual processes in a manner that is natural and familiar to the user, while at the same time being able to manage and update business processes in real time. Reminiscent of the heated battle over the Windows desktop a few years ago, the battle for corporate IT will again be fought at the desktop, but this time we will be talking about workflow productivity tools versus advertising space, and how well a particular BPM solution can seamlessly integrate customer and employee into a cohesive whole.

BEA has taken the lead in this space by being the first to come out with an integrated development environment, centered on BEA WebLogic Workshop, that allows you to easily interface business process orchestration and portal management while providing the necessary Java-based APIs to glue it all together. This is essentially what BEA WebLogic Platform 8.1 is all about, and where the battle lines will be drawn for BEA WebLogic Server 9.0 and beyond.

AUTHOR BIO...

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

CONTACT: joe@sys-con.com

BEA WebLogic DEVELOPER'S JOURNAL

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

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JOE MITCHKO

PRODUCT REVIEW EDITOR
JASON SNYDER

EXECUTIVE EDITOR
GAIL SCHULTZ

EDITOR
NANCY VALENTINE

ASSOCIATE EDITORS
JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR
JENNIFER VAN WINCKEL

RESEARCH EDITOR
BAHADIR KARUV, PhD

WRITERS IN THIS ISSUE
LABRO DIMITRIOU, WES HEWATT, PETER HOLDITCH, ANWAR LUDIN, PRAKASH MALANI, JOE MITCHKO, ROBERT PATRICK, VIMALA RANGANTHAN, VADIM ROSENBERG, AJAY VOHRA, DEEPAK VOHRA

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

PRESIDENT AND CEO
FUAT KIRCAALI

VP, BUSINESS DEVELOPMENT
GRISHA DAVIDA

GROUP PUBLISHER
JEREMY GEELAN

PRODUCTION CONSULTANT
JIM MORGAN

ART DIRECTOR
ALEX BOTERO

ASSOCIATE ART DIRECTORS
LOUIS F. CUFFARI • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR
TAMI BEATTY

SENIOR VP, SALES & MARKETING
CARMEN GONZALEZ

VP, SALES & MARKETING
MILES SILVERMAN

ADVERTISING SALES DIRECTOR
ROBYN FORMA

DIRECTOR OF SALES AND MARKETING
MEGAN MUSSA

ADVERTISING SALES MANAGERS
ALISA CATALANO • CARRIE GEBERT

ASSOCIATE SALES MANAGERS
KRISTIN KUHNLE • BETH JONES

PRESIDENT, SYS-CON EVENTS
GRISHA DAVIDA

NATIONAL SALES MANAGER
SEAN RAMAN

FINANCIAL ANALYST
JOAN LAROSE

ACCOUNTS RECEIVABLE
CHARLOTTE LOPEZ

ACCOUNTS PAYABLE
BETTY WHITE

VP, INFORMATION SYSTEMS
ROBERT DIAMOND

WEB DESIGNERS
STEPHEN KILMURRAY • CHRISTOPHER CROCE

ONLINE EDITOR
LIN GOETZ

CIRCULATION SERVICE COORDINATORS
SHELIA DICKERSON • EDNA EARLE RUSSELL
LINDA LIPTON

JDJ STORE MANAGER
BRUNILDA STARAPOLI

EDITORIAL OFFICES
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9638
SUBSCRIBE@SYS-CON.COM
BEA WebLogic Developer's Journal (ISSN# 1535-9581)
is published monthly (12 times a year)
Postmaster: Send Address Changes to
BEA WEBLOGIC DEVELOPER'S JOURNAL,
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

WORLDWIDE NEWSSTAND DISTRIBUTION
Curtis Circulation Company, New Milford, NJ

NEWSSTAND DISTRIBUTION CONSULTANT
Brian J. Gregory /Gregory Associates/W.R.D.S
732-607-9941 - BJG Associates@cs.com

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

SYS-CON MEDIA

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

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

Not any more.

Get Wily.™

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

wily technology

An Architectural Blueprint

MERGING SOA AND BPM

BY **LABRO DIMITRIOU**

AUTHOR BIO

Labro Dimitriou is a BPMS subject matter expert and grid computing advisor. He has been in the field of distributed computing, applied mathematics, and operations research for over 20 years and has developed commercial software for trading, engineering, and geoscience. Labro has spent the last five years designing BPM-based business solutions.

CONTACT...

labro@bpmsinc.com

Service-oriented architecture (SOA) has become the single most important theme in software engineering. Clearly, the proliferation and unanimous acceptance of Web services, together with a new wave of case-like IDEs that support the development of SOA-based solutions, make SOA the preferred blueprint for building enterprise-wide distributed applications. At the same time, business process management (BPM) is making a strong comeback as a key enabler for modeling and operating the new agile enterprise. Infrastructure vendors have made BPM a key component of their product stack offerings, niche vendors provide vertical business solutions using proprietary BPM systems, and pure-play BPM vendors are gaining wider acceptance.

Although there are indications of the emergence of both trends, no clear and prevailing thought exists about the convergence of the two trends. Are they complementary notations, do they overlap, how do I use them together, and if so is there any additional benefit? Furthermore, why is the third wave of BPM poised to succeed when the BP reengineering of the late '80s failed?

In this three-article series, I will address these questions. First, I discuss how a best-practices architectural blueprint merges service-oriented architecture with a BPM framework in order to provide repeatable ways for modeling and building robust, enterprise-wide integration solutions. I describe why today, more than ever, any enterprise that uses technology to support its mission statement needs to have an architectural blueprint in place. And finally, I discuss what the challenges of doing business in real time are and how a BPM approach can be the key enabler for organizational agility, intelligent enterprise modeling, systems development, and customer-centric operational excellence.

In the second article, I will apply BPM techniques to model and architect a software solution that supports an "apply for car insurance" business scenario. I will cover two designs: a pure BPM and a hybrid. I will also cover some of the emerging modeling tools and standards, and discuss some of the challenges around modeling and various architectural choices and strategies.

In the third and final article, I will use BEA's 8.1 platform to build a running proof of concept. I will discuss the new visual programming paradigm that BEA's IDE introduces, its strengths and weaknesses, and some of the techniques required to build fully distributed, industrial-strength applications. I will also explain why the popular request/reply WEB protocol is at odds with event-based process modeling and its implications in making architectural decisions.

Architectural Patterns – Who Needs Them?

Is software engineering art or science? In science we have unambiguous definitions, theorems, and proofs. In the arts we have tools and techniques, trends, and best practices. In science, postulates are set forward and some become theorems, some are validated after centuries of research, and some may never be answered. In the arts, new technologies bring new trends, like the press and digital photography. If software engineering is a science, it should not be a difficult thing to define the terms we all use in our day-to-day business language, like service, Web service, service-oriented architecture, BPM, and BPM systems (BPMS). Surely I can prove with mathematical rigor the right execution plan in a database query algorithm. But can I answer in a crisp, concise, and universally accepted manner that B2B integration in J2EE is the right answer as opposed to an all .NET Web services solution? I understand that to some of us this is not even a question. Finally, a random survey of any of the usual trade publications points out that everybody is making their own definitions and some are even questioning the very essence of the existence of IT. I have to conclude that software engineering is still more of an art than science. That is exactly why we need best practices, frameworks, and repeatable processes in place.

Patterns encapsulate best practices, define a domain problem in a concise way, describe the forces that make the problem worth identifying, and propose a solution. Patterns do not solve unique problems. Practitioners combine patterns to solve more complex and some times unique problems. Christopher Alexander says, "the pattern is at the same time a thing, which happens in the world and the rule which tells us how to create that thing, and when we must create it. It is both a process and a thing." I recall my all-time favorite definition: an object is a data structure with an attitude or data and behavior. For now, Web services can be viewed as objects with one method. Conversational Web services, as implemented by the BEA WebLogic Platform 8.1, look more like real objects: instantiate it once and keep executing methods. In case you are still not sure that Web services are coarse-grained objects, consider this: (1) IBM, BEA, and Microsoft announced the WS-Eventing specification. It's just like the good old observer pattern for objects. (2) The Open Grid Services Architecture implements Web services' interface inheritance for the grid services protocol. Therefore, Web services provide data and behavior (the thing and the rule in Alexander's definition), and BPMS implements the process component of the pattern. SOA is an architectural pattern addressing enterprise integration and systems development.

SOA is not a revolution but rather an evolution of architectural trends we have seen developing for some time. It evolves around building distributed systems for the enterprise. Web services provide the underlying technology for solving the systems connectivity challenge, admittedly, in a universally accepted and unambiguous way. Perhaps for the first time Web services successfully solves the interoperability challenge, something that CORBA, COM, DCOM, and RPC could never have dreamed of. I am sure XML, acting as the neutral ground, has a lot to do with it. However, the inclusion of the BPMS framework within SOA is a new and revolutionary element. The third wave, described by Howard Smith and Peter Fingar, is a revolutionary new set of concepts, frameworks, and mainstream products. It is dramatically changing the way enterprises are transformed to agile managed and run global and collaborative e-commerce entities.

Business process management has been around for some time, and more so in industries that are not IT related. Concurrent engineering and six sigma were developed to address on-time collaboration in manufacturing and process improvement, and did have quite a success. However, in the late '80s business process reengineering management had rather limited success for many reasons. But the fundamental reason was that reengineering was a paper exercise. No software was around to support such a complex undertaking. BPM engineered the adaptive enterprise without regard to IT systems. As David Taylor writes:

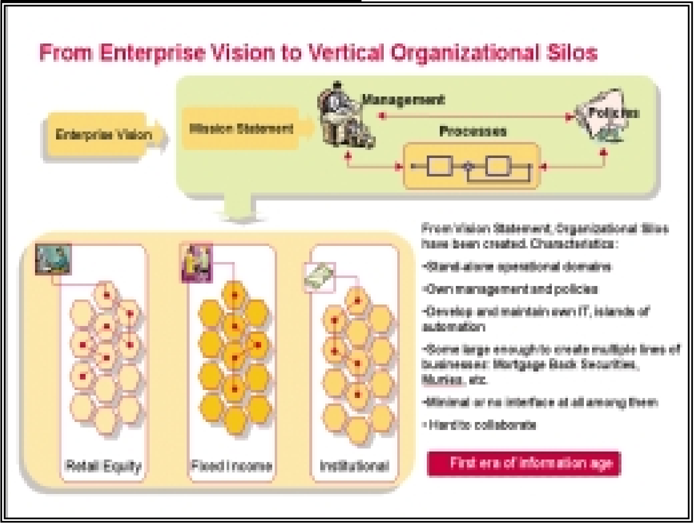
"The demand for continuous process optimization requires a radical rethinking of how information systems are designed and constructed. It is no longer sufficient to produce fixed solutions to fixed problems."

Information systems, like the business models they support, must be adaptive in nature.

Taylor proposed convergent engineering, an OO-based development technique, as a way to develop the adaptive IT. However, OOP could not successfully address distributed computing and enterprise integration. In addition, business analysts, responsible for modeling the enterprise, did not speak OO.

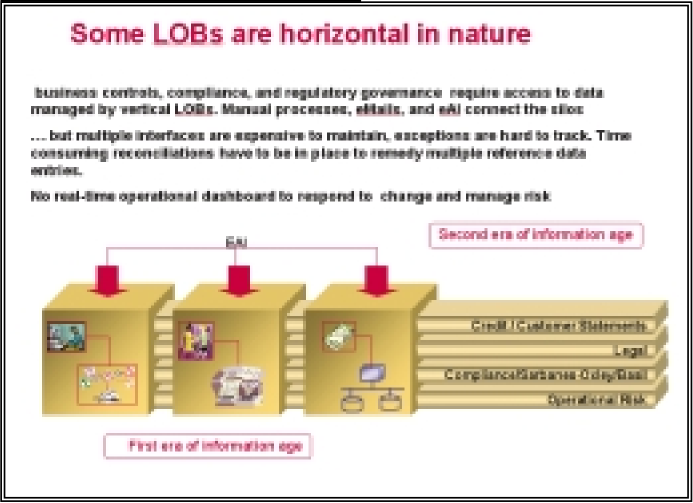
BPMS establishes the process as the unifying construct for modeling, software design, and runtime execution. In the past,

FIGURE 1


From Enterprise Vision to Vertical Organizational Silos
Enterprise Vision
Mission Statement
Management
Policies
Processes
Retail Equity
Fixed Income
Institutional
From Vision Statement, Organizational Silos have been created. Characteristics:
•Stand-alone operational domains
•Own management and policies
•Develop and maintain own IT, islands of automation
•Some large enough to create multiple lines of businesses: Mortgage Back Securities, Munies, etc.
•Minimal or no interface at all among them
•Hard to collaborate
First era of information age

Traditional development stack

FIGURE 2


Some LOBs are horizontal in nature

business controls, compliance, and regulatory governance require access to data managed by vertical LOBs. Manual processes, eMails, and eAI connect the silos

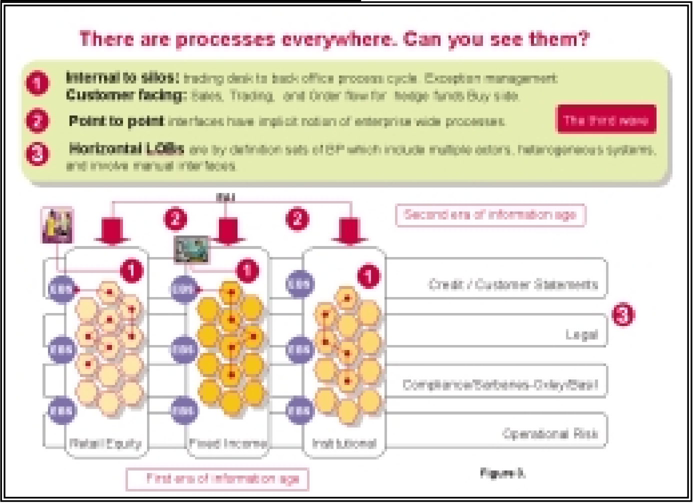
.... but multiple interfaces are expensive to maintain, exceptions are hard to track. Time consuming reconciliations have to be in place to remedy multiple reference data entries.

No real-time operational dashboard to respond to change and manage risk

Second era of information age
EAI
Credit / Customer Statements
Legal
Compliance/Sarbanes-Oxley/Basel
Operational Risk
First era of information age

Horizontal LOBs

FIGURE 3


There are processes everywhere. Can you see them?

1 Internal to silos: trading desk to back office process cycle. Exception management
Customer facing: Sales, Trading, and Order flow for hedge funds Buy side

2 Point to point interfaces have implicit notion of enterprise wide processes.

3 Horizontal LOBs are by definition sets of BP which include multiple actors, heterogeneous systems, and involve manual interfaces

The third wave
EAI
Second era of information age
Credit / Customer Statements
Legal
Compliance/Sarbanes-Oxley/Basel
Operational Risk
Retail Equity
Fixed Income
Institutional
First era of information age
Figure 3.

Interactive processes

development trends have influenced the way we model the enterprise. Functional programming brought functional requirements techniques into vogue. Relational databases brought the proliferation of RDBS analysis and design. Object-oriented programming paved the way for OO analysis and use-case development. But in most cases, business analysts do not speak the development lingo, thus creating the need for another translation with the usual impact in requirements traceability.

BPM specifications are evolving rapidly into standards. Already, there are products in the market place that support process modeling, optimizations, and runtime execution. The BPMS process-centric approach to the systems development life cycle, as implemented by BEA's WebLogic Platform 8.1 and other BPMS products, eliminates the business requirements to runtime impedance mismatch.

Agile enterprise is about adaptive business and adaptive IT systems. If one challenge is new in building enterprise solutions, it is the rate at which requirements change. It is faster than ever. BPMS engines add a new layer in the traditional development stack (see Figure 1) and bring quality of services addressing fundamental challenges in enterprise integration. BPMS engines facilitate the soft wiring of the most volatile parts of the programming: the integration points. Soft wiring is explicitly described in a formal language and executed by BPMS engines (a.k.a., finite state machine engines). Business and IT resources together can review and modify processes in a visual intelligent IDE, as implemented by BEA WebLogic Integrator and other BPMS products. Deployment to runtime BPMS execution engines is one click away. Business simulation can run and performance engineering can be done before completion of the systems; it sounds just like case tools and in a way it is. SOA and BPMS tools bring real-time executive dashboards of the agile enterprise to the mainstream .

In the rest of this article, I'll describe the development of a typical financial services enterprise and propose a migration path to a BPMS-based SOA. The path is incremental but it does require strategic thinking and commitment to the future vision. In return, it will allow early return on investment and transform the legacy to a fully adaptive agile enterprise.

From Enterprise Vision to Organizational Silos

Enterprises start with a vision. CEOs and boards of directors take the vision and craft mission statements. C-level managers define policies and put processes in places to manage execution (see Figure 1). Functional roles and responsibilities are defined and organizational boundaries are created. Lines of business (LOB) can be horizontal or vertical in nature (see Figure 2). Vertical LOBs have the following characteristics:

- Stand-alone operational domains
- Own management and policies.
- Develop and maintain own IT – islands of automation
- Are large enough to create multiple lines of business; for example, mortgage-backed securities, munies, money markets, etc.)

Horizontal LOBs have a different set of characteristics:

- Provide business controls
- Regulatory governance and compliance
- Require access to data managed by vertical LOBs
- Manual processes and paper reports in place

In the second era of information (not to be confused with the second wave) we used various programmatic techniques to link the island of automation starting with FTP, database replication, EAI, and messaging. This approach created a whole new set of problems:

- ***Multiplicity of interfaces:*** A Morgan Stanley Dean Witter report reveals that the typical financial services customer maintains 6,000 interfaces at a cost of $25M annually, and each year builds 900 new point-to-point interfaces at a cost of an additional $25M to build and another $4M to maintain.
- ***Reconciliation processes:*** Must be implemented at each silo, consuming valuable time and expensive resources. This is a common technique for verifying the reference data modified by multiple entities
- ***Processes:*** Hard-wired within the middleware. The time and money spent to capture the processes during analysis is wasted. The most important asset of the enterprise, the process, is buried in a maze of *n*(*n*-1) spaghetti interfaces.
- ***Developing a new horizontal process:*** Requires coordination of multiple LOBs.
- ***Implementing to specific and proprietary interfaces:*** Requires

diagnoSys

Learn how to improve your
Web applications at H&W's
free Web seminar.
Visit www.hwcs.com/wldj1.asp

You might not even know you have a leak, but that little drip can turn into a big deal if it's left unattended. Performance problems with your Web applications work the same way.

Hung threads, memory leaks, and faulty network connections can cause your application to perform poorly or even fail when it matters most – in production. Customers click away and profits plummet. But how do you find the problems if you don't even know they are there?

DiagnoSys is a complete software solution that allows you to analyze the performance of your Web applications through all phases of the life cycle. With true statistical correlation, in-depth analysis, and root cause identification, it helps your teams work together to find problems while they are still little.

The end result? With DiagnoSys, you build vital applications that optimize uptime and keep the bottom line healthy. No problem.

H&W
www.hwcs.com | 1.800.338.6692

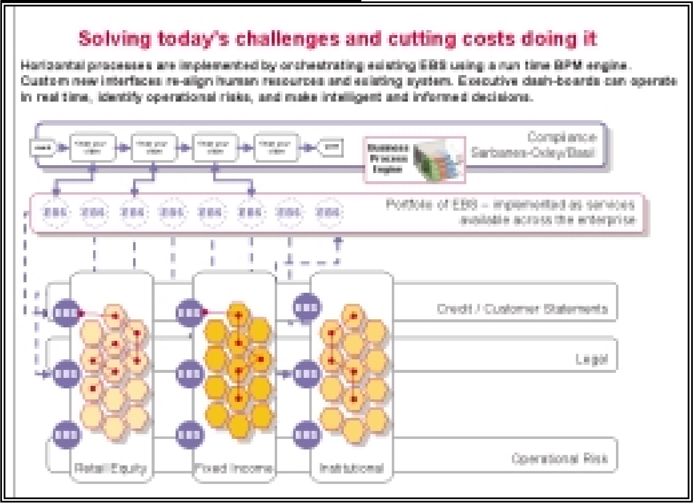
Bridging the Enterprise Computing Gap Since 1979 specialization and one-off programming. Reuse is lost and maintenance increases dramatically.

- ***Exceptions are hard to track:*** Resolution of errors usually requires access to multiple systems. Human intervention and interpretation is inevitable. Finding answers takes a lot of valuable time and has a direct impact on the bottom line in terms of customer satisfaction and profitability.

There Are Processes Everywhere. Can You See Them?

Processes can be customer facing or internal to the organization, or can be part of a larger process. We find internal processes within the same organization. Processes usually involve interaction among people and systems or just interaction among systems (see Figure 3). Trade processing is a good example of a large-scale process. A trader at the front office receives in his sales order system a trade execution order from a hedge fund manager, or he receives a fax or a phone call. The trader checks the inventory system for securities or funds and executes the trade with his counter-party. A paper ticket may be produced and a trading assistant may have to enter it in the downstream system.

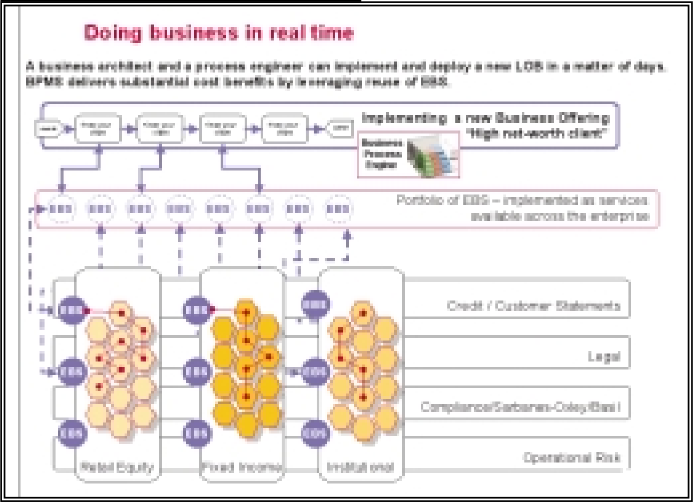
FIGURE 4


Solving today's challenges and cutting costs doing it

Horizontal processes are implemented by orchestrating existing EBS using a run time BPM engine. Custom new interfaces re-align human resources and existing system. Executive dash-boards can operate in real time, identify operational risks, and make intelligent and informed decisions.

EBS enabled

FIGURE 5


Doing business in real time

A business architect and a process engineer can implement and deploy a new LOB in a matter of days. BPMS delivers substantial cost benefits by leveraging reuse of EBS.
Implementing a new Business Offering
"High net-worth client"
Business Process Engine
Portfolio of EBS – implemented as services available across the enterprise
Credit / Customer Statements
Legal
Compliance/Sarbanes-Oxley/Basel
Operational Risk
Retail Equity
Fixed Income
Institutional

Parallel infrastructure

Another internal process starts when a help desk analyst receives an exception report because one of the downstream systems made a wrong re-entry. He then looks up the data in one of the internal blotters (record of original entry), requests a fax from the back office (we assume the trade in question is past settlement date), and perhaps he repeats the same activity again because he hasn't received an answer in two days. The process eventually terminates when the analyst resolves the issue, unless of course he is transferred to another department or outside the company. Then consultants have to come in and trace the problems and processes, usually for a lot of money.

Monthly customer statements are a classical example of a periodic enterprise-wide process, usually owned by a horizontal LOB. In most cases, customers have accounts in products that are supported by different LOBs, for example, equities, U.S. bonds, and foreign currency. It would be rather confusing to send multiple statements at the end of the month. Legal and compliance issues also require cross-referencing multi-silo data. A major challenge of the Patriot and Sarbanes-Oxley Acts (a new business process, albeit not money making) is accessing data owned by a number of LOBs, sometimes halfway around the world. EAI techniques and messaging attempted to address these issues with the limitations explained earlier.

The Way to Agility: BPM-Centric SOA

Let's consider how a BPM-centric SOA with Web services can transform the existing legacy enterprise into an adaptive enterprise. Horizontal processes and exception management are perfect candidates for SOA enablement and can demonstrate a sizable and quick ROI. Without going through the rigors of a business process reengineering, we have to define a good process map. A process map is also the first major step in a business process redesign exercise. Using BPMS design tools (Proactivity, Intalio, Interfacing Technologies), you can associate metrics to processes and activities, for example, performance, cost, IT resources, FTEs, elapsed time, volumes, and so on. Many BPM design tools allow you to run simulations and continue with process optimization (run what-if scenarios) but that's not the focus of this article. For our focus, here are some characteristics of a good process map:

- ***Think processes not functions:*** Processes tell you what work is done and how. Functions describe who does it and where.
- ***Take a customer's point of view:*** Consider processes that start with external business events, e.g., a trade, an order, a claim, a request for price.
- ***Classify customers in a broader sense and based on different quality of services:*** Performance, suppliers, business partners within your ecosystem.
- ***Processes reflect state changes:*** Trade order, cash-to-payment. Start with a manageable number of processes, 6–10. Remember, most people can only retain up to seven things from a page.
- ***Define core processes and subprocesses:*** There are no scientific theories here, just best practices. However, beware of P-calculus2 and Petri-nets; they will bring scientific rigor to BPM within the next 10 years.
- ***Decompose the processes into activities***

The next objective is to define small units of work by decomposing the activities. We call these Elementary Business Services (EBS). If you recall from multidimensional vector algebra, any point in space can be defined as a linear combination of the unit vectors. In our case, we define the universe of all EBS in a way that

Compuware

OptimalJ®

THE POWER TO Develop, Transform, Reuse

Put your J2EE™ application development into overdrive—with unmatched quality and unprecedented flexibility—using Compuware OptimalJ. Seamlessly instill quality into development. Forge existing infrastructures and new technologies into an integrated whole. Realize business agility with applications that are truly reusable, time and time again.

The power is right here.

The leader in IT value.

COMPUWARE®
www.compuware.com

any process can be constructed by orchestrating a subset of EBS. As you might have guessed, we implement EBS as Web services. Identifying the right set of EBS and level of granularity is important. It is as important as it is designing objects. No surprise that the same rules and techniques apply: encapsulation, state dependency, cohesion, loose coupling, and refactoring.

The portfolio of EBS delivers a number of real benefits:

- It is the ultimate guide to reuse. EBS can be orchestrated in any imaginable way to form a new LOB.
- Continuous process improvement does not have to wait for IT to adapt to the new business model.
- EBS are available to the enterprise and business partners within the business ecosystem.
- Retiring a system does not have to be a disruptive process but an incremental one.
- Merger and acquisitions of IT can be achieved in a manageable cost-effective way.
- A new business process can be designed and executed in near real time.

As you can see in Figure 4, we can make EBS available within an instance of BEA WebLogic Platform 8.1 (integration component). Technically, in BEA WebLogic Integration a Web service is called a business-process resource. Using the IDE we orchestrate a new process, add a UI using a portal, and then deploy it as a set of EJBs for execution. It's that simple! The process is now an IT asset, like the database tables, the stored procedures, the legacy COBOL books, and the proprietary computational c libraries.

Many financial services institutions have a horizontal line of business that manages high net worth private clients. In a BPMS SOA-enabled enterprise, developing the IT infrastructure to support such a new LOB can be completed literally in parallel while the business model is put in place (see Figure 5).

Consider the Amazon.com phenomenon, they did not invent any new EBS. All the EBS were in place in any other mail order catalog book store: order book(s), check inventory, charge credit card, print statements, prepare shipment, mail to customer. But it did invent a new process and challenged the establishment without even breaking a sweat.

As Howard Smith and Peter Fingar said: "In the third wave of BPM, stovepipe thinking and point-to-point technical integration give way to flexible, business process-based architectures." Furthermore, Gartner Group is now saying that companies that continue to hard-wire business logic into software or middleware or insist on manual steps will lose out to competitors that deploy process management architectures.

"The proliferation and unanimous acceptance of Web services, together with a new wave of case-like IDEs that support the development of SOA-based solutions, make SOA the preferred blueprint for building enterprise-wide distributed applications"

Doing Business in Real Time

It's hard to predict the future, to say the least, but we are very scientific about it and try to do it all the time, right or wrong. The science of statistics and forecasting is about predicting the future. Portfolio valuation and actuarial study is about projections. In reality, our predictions are based only on past performance and trends we've experienced. Doing business in real time requires anticipation of future business conditions. However, the fundamental business protocols and frameworks have to be in place. Today, technology innovation, BPMS, and SOA are the foundation that aligns business objectives with IT. Processes provide a new layer that encapsulates the change. In the same way that PowerBuilder and VB-like tools proliferated the development of client/server and relational database systems in the early '90s, BPMS engines will establish the future process-driven enterprise. As a matter of fact, I predict that in our life span, we will see a demand for runtime processes to change in real time or requirements for self-modifying processes. Clearly, humans want be able to be in charge of that change but BMPS engines will be able to facilitate it, using UDDI-π (π for process) to find the best possible service contract and make decisions using rules that describe domain expertise and market conditions. With the proliferation of BPMS, agility will give way to extreme adaptation.

Summary

In this article I presented a blueprint for merging SOA and BPM. Starting with a top-down process map of the enterprise we defined the portfolio of elementary business services. Vertical LOBs own and deploy the EBS. Web services implement and make them available to the enterprise. Using an instance of a BPMS engine, a new business process can be designed, developed, tested, and add business value in days, by combining existing EBS.

In my next article I will: (1) cover BPM techniques for modeling a real-world business insurance process, and propose a pure BPM and a hybrid solution; (2) design EBS and implement those using Web services and JMS connectivity; (3) propose a physical architecture using WebLogic Platform 8.1; and (4) discuss BPMS challenges and emerging patterns within a service-oriented architecture.

Until then: processes are everywhere. Can you see them?

References

- Carr, Nicholas G. (May 2003). "IT doesn't Matter". *Harvard Business Review*.
- Alexander, Christopher. (1979). *The Timeless Way of Building*. Harvard University Press.
- Gamma,E.; Helm, R.; Johnson, R.; Vlissides,J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Smith, Howard, and Fingar, Peter. (2003). *Business Process Management: The Third Wave*. Meghan-Kiffer Press.
- Shina, Sammy. (1991). *Concurrent Engineering and Design for Manufacturing of Electronics Products*. Van Nostrand Reinhold.
- Pande, Peter S., et al (2000). *The Six Sigma Way*. McGraw-Hill Trade.
- Taylor, David A. (1992). *Business Engineering with Object Technology*. John Wiley & Sons, Inc.
- Morgan Stanley Dean Witter, (April 2000). "The B2B Internet Report".
- Gartner Group (November 2001). "Business Process Management – Are you experienced?"

Optimize J2EE.

The J2EE revolution is here to increase performance, value, and lower IT costs.

It's a solution from Mercury Interactive that makes your whole J2EE applications ecosystem work right.

It's the fastest way to find the toughest J2EE problems from the development to the live applications. Even pinpoints root cause and the line of source code.

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

It's the Business Technology Optimization revolution.

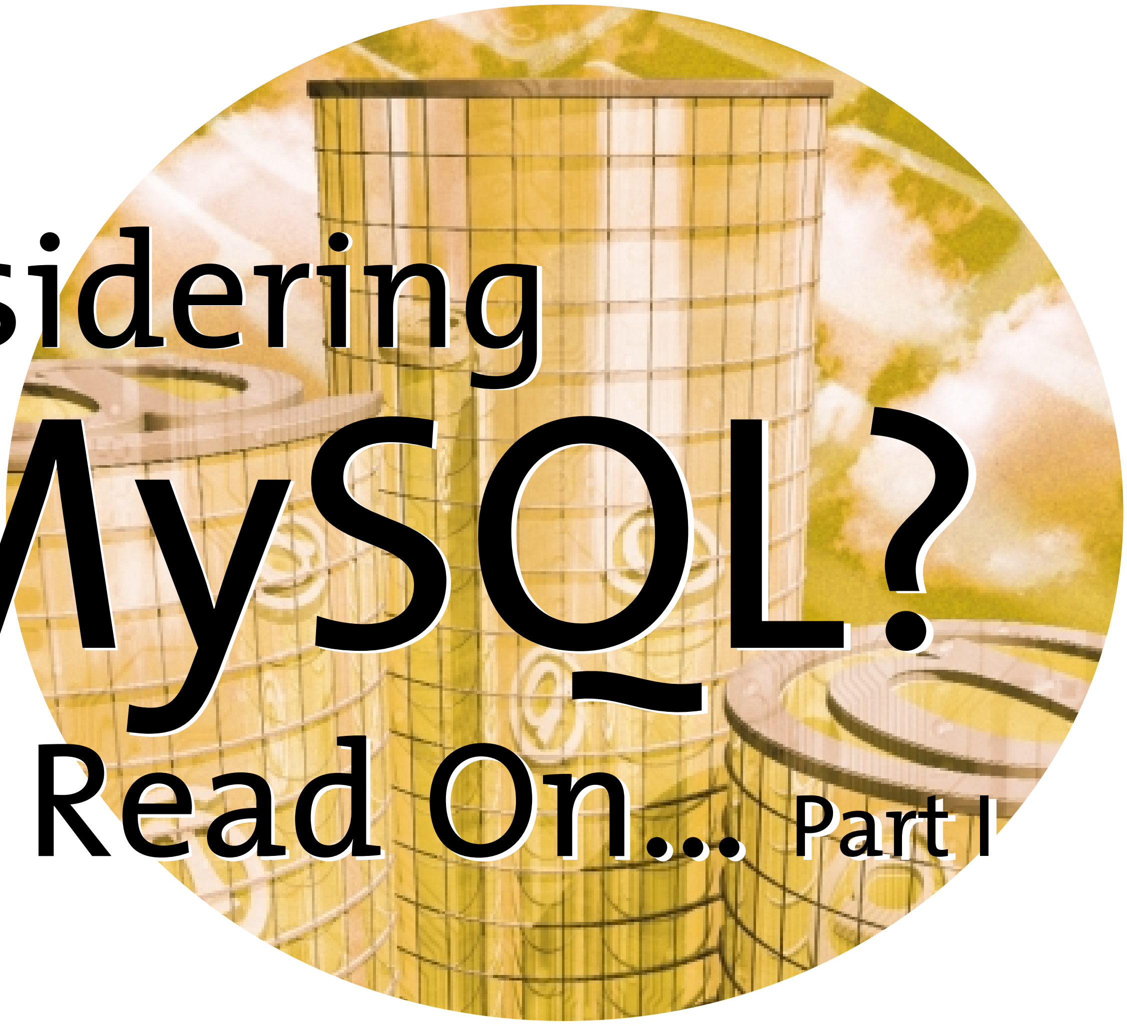
And Mercury is the only one bringing you a revolutionary solution for J2EE.

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

at **www.mercuryinteractive.com/optimizej2ee**

Get Optimized.™

MERCURY INTERACTIVE®



Considering MySQL? Read On... Part I

A POWERFUL COMBINATION CREATES A STRONG ARCHITECTURE

BY **PRAKASH MALANI**

AUTHOR BIO...

Prakash Malani has extensive experience in architecting, designing, and developing object-oriented software and has done software development in application domains such as entertainment, retail, medicine, communications, and interactive television. He practices and mentors leading technologies such as J2EE, UML, and XML. Prakash has published various articles in industry-leading publications.

CONTACT...

pmalani@ebuilt.com

MySQL is a small, fast, and efficient database. This article discusses leveraging MySQL as the database with BEA WebLogic Server 8.1.

We will look at using MySQL as the database engine where the application is developed using BEA WebLogic Workshop 8.1 and deployed to BEA WebLogic Server 8.1. Using an archetypical Java 2 Enterprise Edition (J2EE) architecture, I will evaluate the impact of using MySQL from various aspects such as choosing the correct version of MySQL, setting up the server, and making development adjustments. The impact on development and deployment of bread-and-butter technologies such as Enterprise JavaBeans (EJBs), Java DataBase Connectivity (JDBC), Java Message Service (JMS), and the Java Transaction API (JTA) are evaluated. Various pitfalls are uncovered, logically approached, and methodically solved. The information presented here will not only enhance your understanding of the tools and technologies utilized, but also save you countless hours. Even readers who employ different database technologies will find the information and material practical and useful.

Introduction

The development tool of choice is BEA WebLogic Workshop 8.1. I'll describe an archetypical J2EE architecture and explore the impact of the decision to use MySQL with J2EE technologies such as JDBC and JMS. Part 2 of this series will evaluate the impact of EJBs, the core component model of J2EE as well as the JTA. The article describes configuration and development changes, adjustments, and modifications.

Architecture

The sample application has a standard J2EE architecture consisting of a database tier, an application tier, and an interface tier as depicted in the Figure 1. The database, or back-end, tier consists of MySQL as the Relational DataBase Management System (RDBMS). The application tier is BEA WebLogic Server 8.1. The application server includes a JMS server and an EJB container. The JMS server hosts destinations such as a queue utilized by the sample application for asynchronous processing. The sample application consists of many EJBs that leverage container managed transaction demarcation (CMTD). Tables in the database are mapped to entity EJBs with container-managed persistence (CMP). The entity EJBs are fronted with a session EJB facade. There is a Message-Driven Bean (MDB) that listens to a queue and processes messages. Other applications, such as command-line applications, rich graphical user interface (GUI) applications, or Web applications, leverage the EJB components through the facade. They exchange data with the session facade using value objects (also known as data transfer objects). The value objects corresponding directly to the entity beans are automatically generated by WebLogic Workshop.

Choosing the "Right" MySQL

MySQL comes in various shapes and forms and has at least four different incarnations: MySQL Standard, MySQL Max, MySQL Pro, and MySQL Classic. MySQL Standard and MySQL Pro are identical except for the license. MySQL Standard is licensed under the GNU Public License (GPL), whereas MySQL Pro is a commercially licensed version of MySQL Standard. MySQL Max includes cutting-edge and experimental features and is not recommended for production use. MySQL Classic is available only under commercial license and excludes important features. For this application, support is required for critical features such as transactions and referential-integrity (i.e., foreign key) constraints. The choice of MySQL among its various incarnations is either MySQL Standard or MySQL Pro. *Note:* Any further reference in this article to MySQL implies usage of MySQL Standard. (More information about the different types of MySQL is available at www.mysql.com/products/mysql/index.html.)

MySQL presents a number of choices for table types, each offering features that have their own pros and cons. The table types are ISAM, MyISAM, HEAP, MERGE, BDB, and InnoDB. Many factors determine the choice of a table type. These factors include, but are not limited to, performance, transactions, row-level locking, and crash recovery. However, the crucial features for the sample application are transactions and referential-integrity constraints.

The InnoDB table type is the only one that meets the criteria. There are at least two ways to specify an InnoDB table type. One is to start the MySQL database server using --default-table-type=InnoDB. A table created with this option is of InnoDB type. (*Note:* If the default table type of InnoDB is not specified, the default table type is MyISAM.) The other way to specify InnoDB table type is to explicitly mention table type in the Data Description Language (DDL) of the create table script. (More information about MySQL table types is available at www.mysql.com/doc/en/Table_types.html.)

A specific feature about MySQL and foreign constraints is that before the constraint can be created, an index on the column must already exist. For example, consider a one-to-one relationship between the tables Person and Buyer. Buyer has a foreign key to Person. Buyer has a column named Person_Id that is a foreign key to a column named Person_Id in the Person table. But before the foreign key can be established, an index must be created on column Person_Id in the Buyer table. Otherwise, the creation of a foreign key constraint fails. Refer to the sem.sql DDL file in the source code example (the source code is online at www.sys-con.com/weblogic/sourcec.cfm).

The default database privileges are different for different operating systems. For example, the default privileges on Windows give all local users full privileges without specifying a username or password. Therefore, an important validation is being able to connect to the MySQL database server engine. One way to connect is by using the client program that comes with MySQL. Another way is by using JDBC and a program like DbVisualizer. (See the section "Verify Connectivity using DbVisualizer". Information about default privileges is available at www.mysql.com/doc/en/Default_privileges.html.)

The default case sensitivity of table names is based upon the operating system. For example, on Windows the table names are case insensitive. This is especially important if development is done on one platform but deployment is on another. To avoid such problems, one recommendation is to start the server by setting lower_case_table_names=1 variable. (More information on this variable is available at www.mysql.com/doc/en/Name_case_sensitivity.html.)

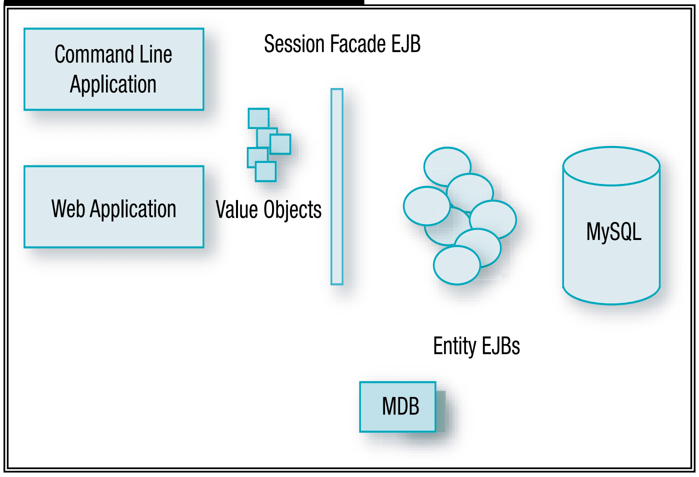
Logging is enabled in MySQL by starting the MySQL database server using the --log option. Starting the MySQL server with logging enabled is highly recommended. Logging enables monitoring of Structured Query Language (SQL) statements executed by MySQL and can be a crucial tool in debugging and trouble-shooting issues. (More information on the MySQL log file is available at www.mysql.com/doc/en/Query_log.html.)

The various options, such as default table type and logging, can be specified in a couple of ways. One is to pass the options in as command-line arguments to mysqld, the MySQL database engine executable. The other option is to specify these options in either a my.ini file or my.cnf option file. (More information on option files, including where to place them, is available at www.mysql.com/doc/en/Option_files.html.)

Before we proceed further, complete the following steps:

1. Download MySQL: www.mysql.com/downloads/mysql-4.0.html.
2. Install MySQL with these instructions: www.mysql.com/doc/en/Installing.html.
3. Specify additional options such as logging and table type.
4. Start MySQL (e.g., using mysqld -- console).
5. Verify connectivity to the server using mysql (e.g., mysql –p –u

FIGURE 1



Archetypical J2EE sample application architecture

root). *Note:* Just press enter when prompted for a password if local on Windows)
6. Create a database named semdb using create database semdb.
7. Use the database semdb using semdb.
8. Load schema using mysql command source and specifying the fully qualified path to sem.sql (e.g., source c:/MySQL_WLS/db/-sem.sql). *Note:* Use forward slashes even on Windows.
9. Using the grant.sql file available in the source code, grant privileges using source fully qualified path to grant.sql. *Note:* Use forward slashes even on Windows.

Downloading the JDBC Driver

JDBC requires a JDBC driver to connect to the database server. Connector/J is the JDBC driver for MySQL. Download and install Connector/J from www.mysql.com/downloads/api-jdbc-stable.html. The download contains a mysql-connector-java-3.0.9-stable-bin.jar file that contains the JDBC drivers.

The following section on verifying the connectivity to MySQL server is optional, but highly recommended.

Verify Connectivity Using DbVisualizer

DbVisualizer ships with BEA WebLogic Platform 8.1. Launch DbVisualizer (e.g., Start Menu > BEA WebLogic Platform 8.1 > Other Development Tools > DbVisualizer on Windows) and add the drivers to DbVisualizer. Using the Connection/J MySQL JDBC driver, connect to MySQL instance (see the section, "Connection to MySQL Database Instance).

Adding JDBC Drivers to DbVisualizer

The following steps describe adding the drivers to DbVisualizer.
1. Select Database Menu.
2. Select DriverManager Menu item.
3. Select Add new driver location toolbar menu item.
4. Select mysql-connector-java-3.0.9-stable-bin.jar.
5. Close.

Connection to MySQL Database Instance

The following steps describe connecting to the MySQL database instance.
1. Select Database Menu.
2. Select Add Database Connection Menu Item.
3. Specify Connection Alias (e.g., sem).
4. Choose JDBC Driver as com.mysql.jdbc.Driver from the drop-down list.
5. Choose Database URL as jdbc:mysql://<host>/<database> from the drop-down list.
6. Change <host> (e.g. localhost).
7. Change <database> to semdb.
8. Specify Userid as sem.
9. Specify Password as sem.
10. Select Connect.

WebLogic Domain Configuration Changes

The BEA WebLogic Domain Configuration Wizard is used to create WebLogic Server domains. A particular type of domain is created based on functionality required. For this particular application, create a WebLogic Workshop domain. For example, create a WebLogic Workshop domain named SEMDomain.

In order to use JDBC with BEA WebLogic Server, WebLogic Server needs to know about the JDBC driver. The JDBC driver needs to be in the CLASSPATH so that the server can locate the driver. For the SEMDomain, add the JDBC driver to the CLASSPATH so that the WebLogic Server can find the JDBC driver.

Add JDBC Driver To The CLASSPATH

The following steps describe adding the Connector/J MySQL JDBC driver to the CLASSPATH.
1. Edit …/SEMDomain/_cfgwiz_donotdelete/startscript.xml.
2. Add mysql-connector-java-3.0.9-stable-bin.jar to the CLASSPATH. Refer tothe startscript.xml file in the source code.
3. Add system property Dweblogic.ejb20.cmp.rdbms.codegen.verbose=true as JAVA_OPTIONS. Refer to the startscript.xml file in the source code.
4. Run the Configuration Wizard to extend the SEMDomain.
5. Select DefaultWebApp as the extension.
6. Finish the Configuration Wizard.

The following section on enabling more WebLogic Server logging is optional, but highly recommended.

WebLogic Logging

By default, BEA WebLogic Server performs only limited logging. During development, logging and observing of more information is very useful. For the SEMDomain, change the logging level to Info.

Changing the Logging Level

The following steps describe changing the BEA WebLogic Server logging configuration.
1. Make sure BEA WebLogic Server is running.
2. Launch the WebLogic Server Console.
3. Log into the Console.
4. Select Servers/cgServer.
5. Select Logging tab.
6. Check Debug to Stdout checkbox.
7. Select Stdout Severity Threshold as Info from the drop-down list.

JMS Domain Configuration Changes

By default, a newly created domain like the SEMDomain uses PointBase as the database. A connection pool named cgPool connects to PointBase and a corresponding datasource named cgDataSource. The JMS server uses the cgDataSource as the persistence store. To use another database, such as Oracle, just change the configuration of the connection pool. The JMS server then uses Oracle as the persistence store. However, simply changing the connection pool to MySQL does not work, because when the WebLogic server boots-up, the JMS server complains that MySQL is not a supported database. A practical alternative is to change from using a database as the persistence store to using a file as the persistence store. WebLogic JMS supports a persistence file store.

"Understanding the impact of leveraging MySQL on technologies such as JDBC and JMS is crucial to successful project implementation"

For the SEMDomain, change the persistence store to a file store.

Changing the File Store

The following steps describe configuring the file store:

1. Create a directory where the file store is going to be placed.
2. Make sure BEA WebLogic Server is running.
3. Launch the WebLogic Server Console.
4. Log into the console.
5. Select Services/JMS/Stores in the left navigation pane.
6. Select Configure a new JMS File Store.
7. Specify Name (e.g., cgJMSFileStore).
8. Specify Directory that was created in the earlier step.
9. Select Services/JMS/Servers/cgJMSServer.
10. Change the Persistence Store (to, for example, cgJMSFileStore) by selecting it from the drop-down list.

The bottom line is that MySQL cannot be used as a persistence store for JMS. Using a file persistence store is a viable alternative. Keep this in mind when creating other JMS servers.

In order to follow the source code example, complete an additional JMS configuration for the sample application before proceeding further:

- Set up a JMS file store named semJMSFileStore.
- Set up a JMS connection factory named semJMSConnectionFactory with jms/semJMSConnectionFactory as the JNDI name.
- Set up a JMS server named semJMSServer with semJMSFileStore as the persistent store.
- Set up a distributed destination named PersonQueue with jms/PersonQueue as the JNDI name.
- Deploy the PersonQueue on the semJMSServer.

JDBC Domain Configuration Changes

To utilize the JDBC API in the BEA WebLogic Server applications, create and deploy a connection pool and a datasource. First, create the connection pool, then create the datasource on top of the connection pool.

Before proceeding further, complete the JDBC configuration for the sample application. Set up the following:

- ***Connection pool:*** Refer to "Configuring the Connection Pool" for detailed instructions.
- ***Datasource:*** Use jdbc/semJDBCDataSource as the JNDI name. Refer to "Configuring the Datasource" for detailed instructions.

Configuring the Connection Pool

The following steps describe configuration of the connection pool.

1. Make sure BEA WebLogic Server is running.
2. Launch the WebLogic Server Console.
3. Log into the console.
4. Select Services/JDBC/Connection Pools.
5. Select Configure a New JDBC Connection Pool.
6. Select MySQL from the Database Type drop down list.
7. Select MySQL's Driver (Type 4) Version: Any from Database Driver.
8. Specify Name (e.g., semJDBCConnectionPool).
9. Specify Database Name (e.g., semdb).
10. Specify Host Name (e.g., localhost).
11. Specify Username (e.g., sem).
12. Specify Password (e.g., sem).
13. Test the driver configuration.
14. Create and deploy.

Configuring the Datasource

The following steps describe configuration of the datasource:

1. Select Services/JDBC/Data Sources.
2. Select Configure a New JDBC Data Source.
3. Specify Name (e.g. semJDBCDataSource).
4. Specify JNDI Name (e.g., jdbc/semJDBCDataSource).
5. Select Continue.
6. Specify Pool Name (e.g., semJDBCConnectionPool).
7. Target the datasource by clicking on Create.

Conclusion

This article discussed how to select the "right" version of MySQL and described various changes to the BEA WebLogic Domain Configuration to support key J2EE technologies such as JDBC and the Java Message Service. My next article will explore advanced topics such as EJBs and JTA.

Understanding the impact of leveraging MySQL on technologies such as JDBC and JMS is crucial to successful project implementation. As illustrated here, MySQL, BEA WebLogic Workshop, and BEA WebLogic Server form a powerful combination to architect, design, and deploy real-world applications.

I want to thank Steve Ditlinger, Roshni Malani, and Sarah Woo for reviewing this article and providing invaluable feedback.

References

- *To discuss the article and ask questions:* http://groups.yahoo.com/group/bartssandbox. Free membership enrollment is required.
- *Main MySQL Web site:* www.mysql.com
- *Starting point for MySQL documentation:* www.mysql.com/documentation/index.html
- www.oreillynet.com/lpt/wlg/3946
- DuBois, Paul (2003). *MySQL: The Definitive Guide to Using, Programming, and Administering MySQL 4 databases.* Sams. An excellent reference. (www.bookpool.com/.x/d4jha9om4m/sm/0735712123).
- *J2EE patterns:* http://java.sun.com/blueprints/patterns/index.html). For the Session Facade Pattern used in the sample application refer to http://java.sun.com/blueprints/corej2eepatterns/Patterns/SessionFacade.html and http://java.sun.com/blueprints/patterns/SessionFacade.html. For the Value Object Pattern used in the sample application refer to http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html and http://java.sun.com/blueprints/patterns/TransferObject.html

Developing Entity EJBs

ENHANCED PERFORMANCE AND DEVELOPMENT

BY **DEEPAK & AJAY VOHRA**

Enterprise JavaBeans (EJBs) are application components that implement the EJB architecture specification and are part of the Java 2 Enterprise Edition (J2EE) platform. EJBs are ideally suited for the development and deployment of distributed, scalable, transactional, secure, portable, component-based business applications.

EJB-based business applications require an EJB container for runtime execution and all J2EE-compliant application servers, including WebLogic 8.1, provide an EJB container.

The principal motivation underlying EJB architecture is separation of concerns: it separates the application infrastructure–related concerns such as transaction processing and security from core application concerns such as business logic. Briefly, the EJB architecture achieves this separation of concerns by specifying a separation of responsibilities between an EJB container and an EJB developer. For example, it is the responsibility of an EJB container to transparently implement transaction processing, and it is the responsibility of an EJB developer to implement business logic. An EJB container may require some hints to do its job, but providing such hints to the container (through XML-based deployment descriptors) is very inexpensive compared to actually implementing these infrastructure-related activities. Overall, this strategy of separation of concerns makes business application development much more efficient than the alternatives.

There are four versions of the EJB architecture specification: 1.0, 1.1, 2.0, and 2.1. For all practical purposes, EJB version 1.0 is obsolete and the latest version, 2.1, is too recent to be widely available. BEA WebLogic Server 8.1 supports both versions 1.1 and 2.0. We strongly recommend EJB version 2.0 while developing in WebLogic Server 8.1.

An entity bean is an EJB. Besides the general motivation behind an EJB, the specific motivation behind using an entity bean is to provide an in-memory, shareable, object-oriented view for a business-domain entity that exists in persistent storage, typically as a row in a table in a relational database. In this article, the specific issues associated with the design, development, and deployment of an Entity EJB in the context of the WebLogic Server 8.1 environment are discussed. For a general tutorial on EJB technology, we recommend the J2EE tutorial at http://java.sun.com/j2ee.

AUTHOR BIO

Ajay Vohra is a senior software engineer with Compuware.

Deepak Vohra is a Web developer, and a NuBean consultant.

CONTACT...

ajay_vohra@yahoo.com, dvohra09@yahoo.com

Overview

Entity beans are designed to manage data in a relational database. Entity EJB development in BEA WebLogic Server 8.1 consists of designing, generating, packaging, and deploying an EJB.

From a design point of view, entity beans can be classified along two orthogonal axes: persistence and access. Along the axis of persistence, there are two types of entity beans: bean-managed persistence (BMP) and container-managed persistence (CMP). In the case of CMP, the EJB container manages the persistence of an entity bean; in the case of BMP, an entity bean developer through specified Java code manages the persistence of an entity bean. The choice between CMP and BMP is mutually exclusive. Along the axis of access, there are two types of entity beans: remote and local. A remote entity bean provides transparency of location and can be accessed from a different Java virtual machine; a local entity bean, by contrast, can only be accessed from within the same application server. The choice between local and remote is not mutually exclusive and one can design beans with a dual interface.

In the EJB architecture specification, each entity EJB component is comprised of a set of specified Java classes and a set of specified XML deployment descriptors. EJB packaging consists of packaging all the specified EJB Java class files and XML deployment descriptor files in a Java Archive (JAR) file. If there are helper Java class files that an entity EJB depends upon, such class files can either be included within the EJB JAR file, or can be packaged separately in a different JAR file.

EJB deployment consists of either directly deploying the EJB JAR file and any dependent JAR files within BEA WebLogic Server 8.1, or first packaging the EJB JAR file and any dependent JAR files within an enterprise application archive (EAR) file, and then deploying the EAR file within WebLogic Server 8.1.

Designing an EJB

The principal design choices in the design of an entity bean are:

1. Whether to design a CMP or BMP entity bean
2. Whether to design a local, remote, or dual interface entity bean
3. Whether to design a coarse-grained or fine-grained entity bean
4. Whether to use data transfer objects or use individual get and set methods to access data from an entity EJB

These design choices for developing an entity EJB are discussed below.

CMP vs BMP

Keeping in mind that there may be rare legitimate exceptions to this rule, in general we strongly recommend using a CMP design. There are three main reasons for selecting a CMP over BMP.

1. Compared to BMP, CMP provides portability across various databases because CMP entity beans do not contain any database-specific persistence code. CMP is easy to design, implement and maintain.
2. In general, CMP has better performance than BMP because the database-specific code is automatically generated by an EJB container and is optimized for the target database.
3. CMP makes it very easy to programmatically navigate through a network of related EJBs, using local interfaces.

Local, Remote, or Dual Interface

Because CMP manages relationships between entity EJBs through a local interface, we strongly recommend always providing a local interface. We think a remote interface is rarely needed, but if careful evaluation demonstrates the need for a remote interface, go ahead and design a dual interface.

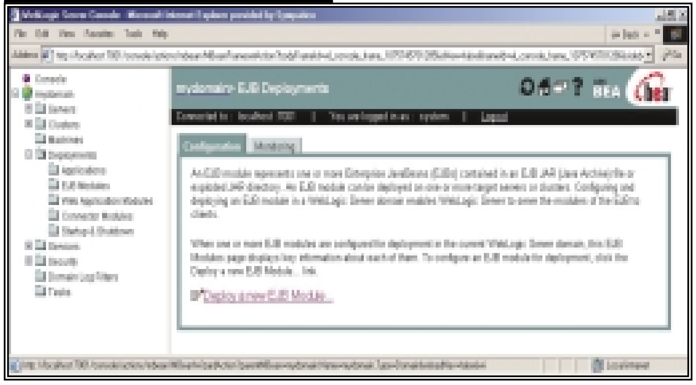
Local interfaces provide optimized access to an EJB by local clients; remote method invocation (RMI) semantics are not required to access an EJB with local interfaces by a local client.

Remote clients, which are located on a different virtual machine than the EJB container, require RMI and remote interfaces to access an EJB. The obvious theoretical disadvantage of solely designing a local entity is that only clients in the same application server are able to access the entity bean. However, this is only a theoretical disadvantage because in practice entity EJBs are rarely accessed from outside an application server.

Coarse Grained vs Fine Grained

This is a very controversial subject so carefully calibrate various opinions on this subject against your personal experience. Our opin-

FIGURE 1


Console
mydomain
Servers
Clusters
Machines
Deployments
Applications
EJB Modules
Web Application Modules
Connector Modules
Startup & Shutdown
Services
Security
Domain Log Filters
Tasks

mydomain> EJB Deployments

Connected to : localhost:7001 | You are logged in as : system | Logout

Configuration | Monitoring

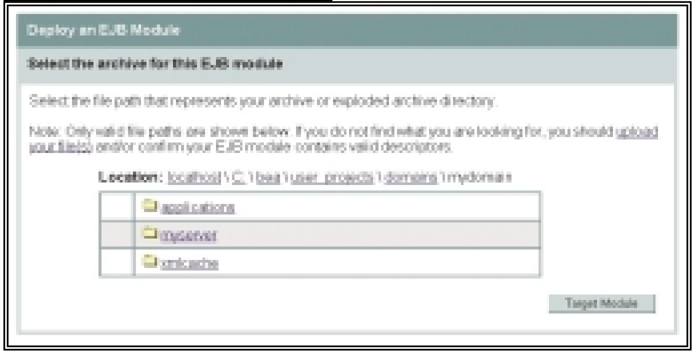
An EJB module represents one or more Enterprise JavaBeans (EJBs) contained in an EJB JAR (Java Archive) file or exploded JAR directory. An EJB module can be deployed on one or more target servers or clusters. Configuring and deploying an EJB module in a WebLogic Server domain enables WebLogic Server to serve the modules of the EJB's clients.

When one or more EJB modules are configured for deployment in the current WebLogic Server domain, this EJB Modules page displays key information about each of them. To configure an EJB module for deployment, click the Deploy a new EJB Module... link.

Deploy a new EJB Module...

EJB Deployments Frame

FIGURE 2


Deploy an EJB Module

Select the archive for this EJB module

Select the file path that represents your archive or exploded archive directory.

Note: Only valid file paths are shown below. If you do not find what you are looking for, you should upload your file(s) and/or confirm your EJB module contains valid descriptors.

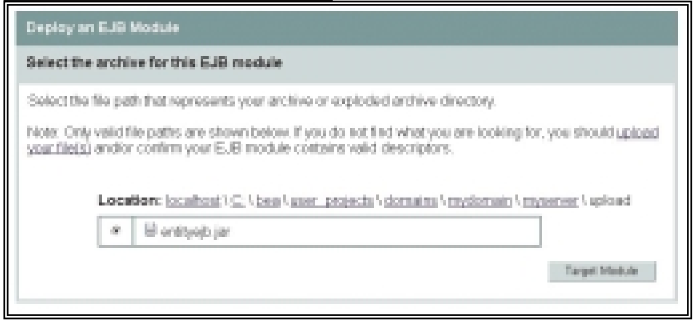
Location: localhost \ C: \ bea \ user_projects \ domains \ mydomain

applications
myserver
xmlcache

Target Module

Deploy an EJB Module Frame

FIGURE 3


Deploy an EJB Module

Select the archive for this EJB module

Select the file path that represents your archive or exploded archive directory.

Note: Only valid file paths are shown below. If you do not find what you are looking for, you should upload your file(s) and/or confirm your EJB module contains valid descriptors.

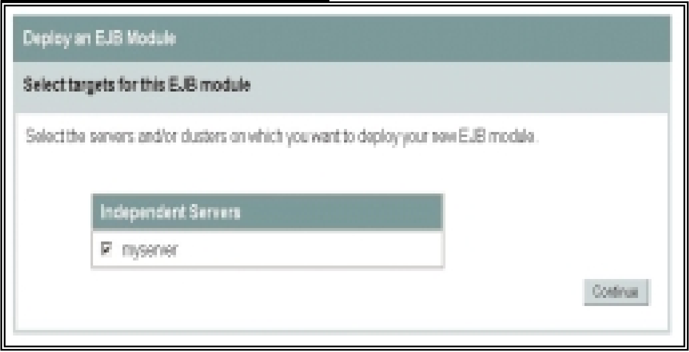
Location: localhost \ C: \ bea \ user_projects \ domains \ mydomain \ myserver \ upload

entityejb.jar

Target Module

Select the archive for this EJB module Frame ion is that entity EJBs are most often used to represent individual entities in an application's business domain, so go ahead and make entity beans as fine grained as possible, but confine the design to only providing a local interface. The controversy around this issue started during EJB architecture specification 1.x versions, when entity EJBs could only be accessed through a remote interface. Any reasonable arguments against using fine-grained entity EJBs formulated during EJB 1.x versions have become obsolete in EJB architecture specification 2.0 and should be deprecated. Please be warned: there are some experts who would disagree with us, so we encourage you to experiment and form your own opinion on this subject.

FIGURE 4


Deploy an EJB Module

Select targets for this EJB module

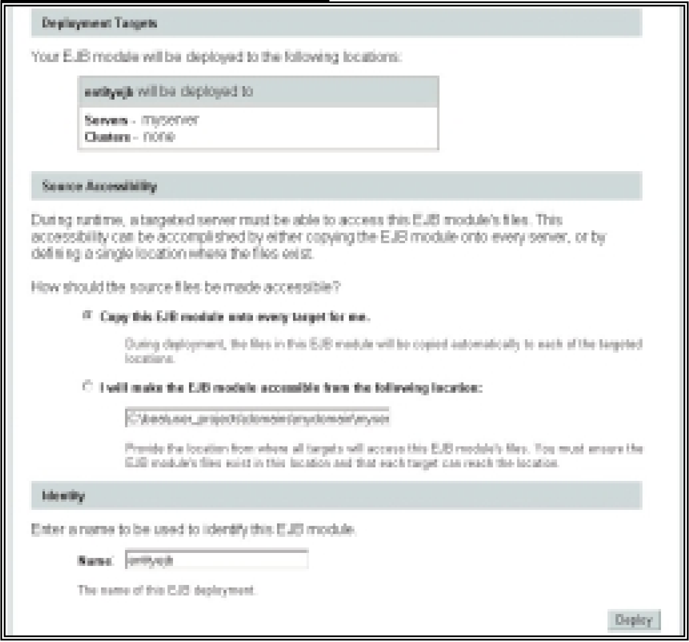
Select the servers and/or clusters on which you want to deploy your new EJB module.

Independent Servers

☑ myserver

Select Targets for this EJB module Frame

FIGURE 5


Deployment Targets

Your EJB module will be deployed to the following locations:

entityejb will be deployed to

Servers - myserver
Clusters - none

Source Accessibility

During runtime, a targeted server must be able to access this EJB module's files. This accessibility can be accomplished by either copying the EJB module onto every server, or by defining a single location where the files exist.

How should the source files be made accessible?

● **Copy this EJB module onto every target for me.**

During deployment, the files in this EJB module will be copied automatically to each of the targeted locations.

○ **I will make the EJB module accessible from the following location:**

C:\bea\user_projects\domains\mydomain\myser

Provide the location from where all targets will access this EJB module's files. You must ensure the EJB module's files exist in this location and that each target can reach the location.

Identity

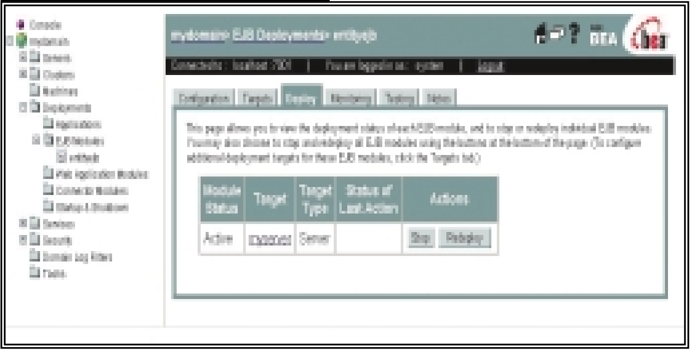
Enter a name to be used to identify this EJB module.

Name: entityejb

The name of this EJB deployment.

EJB Module Name Frame

FIGURE 6



This page allows you to view the deployment status of each EJB module, and to stop or redeploy individual EJB modules. You may also choose to stop and redeploy all EJB modules using the buttons at the bottom of the page. (To configure additional deployment targets for these EJB modules, click the Targets tab.)

Module Status	Target	Target Type	Status of Last Action	Actions
Active	myserver	Server		Stop Redeploy

EJB Deploy Tab

Data Transfer Objects vs Get and Set Methods

This is another controversial subject. Our view is as follows:

- Expose all the get accessor methods for the CMP persistent fields in the local interface of an entity EJB.
- Create wrapper set methods for each CMP persistent field that is not part of an entity EJB's primary key, and expose these wrapper methods in the local interface of an entity EJB. The motivation behind the wrapper methods is that CMP requires all persistent fields to have abstract setXXX methods, and if you need to do any validation within these setXXX methods, you can do so in the wrapper method and then call the corresponding setXXX method. If such validation is not needed, then it is quite appropriate to skip the wrapper methods and directly include the setXXX methods through the local interface.
- In the rare circumstance that you need a remote interface for an entity EJB, define a data transfer object per entity EJB, and expose a get and set method using the data transfer object as a parameter in the remote interface of an entity EJB.

Generating and Packaging an EJB

In the EJB architecture specification, each entity EJB component is comprised of a set of specified Java classes and a set of specified XML deployment descriptors:

1. A required bean class that implements the core functionality of an entity bean
2. A remote interface, local interface, or dual interface that provides an appropriate client view for the entity bean
3. A remote home interface, local home interface, or dual home interface that provides an appropriate interface for life-cycle management of an entity bean
4. In the case of entity beans with a composite primary key, a required primary key class
5. An ejb-jar XML deployment descriptor file, as specified by EJB architecture specification version 2.0
6. A vendor-specific weblogic-ejb-jar XML deployment descriptor file, as specified by BEA WebLogic Server 8.1
7. In the case of a CMP entity bean, a vendor-specific weblogic-cmp-rdbms-jar XML deployment descriptor file, as specified by BEA WebLogic Server 8.1

The example entity EJB used here is a CMP entity EJB with a remote interface and is named AccountEJB. It can be found in the included sample files installed with WebLogic Server 8.1 under /weblogic81/samples/server/examples/src/examples/ejb20/basic/containerManaged directory.

The entity bean EJB classes and interfaces may be generated with the EJBGen tool.

EJBGen

EJBGen is an EJB 2.0 code generator that generates the local/remote, local-home/remote-home interfaces, primary key class, and the deployment descriptors from an EJB bean class. EJBGen tags are used in the EJB bean class to specify the different EJB design configurations (for example, local/remote and CMP/BMP). In WebLogic Server 8.1 SP01, the EJBGen classes are included in /weblogic81/server/lib/weblogic.jar. In WebLogic Server 8.1 SP02 & SP03, the EJBGen classes are included in /weblogic81/server/lib/ejbgen.jar file. Add ejbgen.jar to Classpath to use the EJBGen tool.

EJBGen is invoked with the command:

```
javadoc -docletpath ejbgen.jar -doclet weblogic.tools.ejbgen.EJBGen
<EjbBeanClass>.java
```

Some of its options are:

- ***-d [directory]:*** Directory in which the EJB classes/interfaces and deployment descriptors are created.
- ***-descriptorDir [directory]:*** Directory in which the deployment descriptors are created.
- ***-wls7***: With the –wls7 option WebLogic Serve 7.1 deployment descriptors are generated.

If an earlier version of the BEA WebLogic Server deployment descriptors are to be converted to WebLogic Server 8.1 deployment descriptors, use the DDConverter.

DDConverter

DDConverter is a command-line tool to convert earlier versions of XML deployment descriptors (ejb-jar.xml, weblogic-ejb-jar.xml, and weblogic-cmp-rdbms-jar.xml) to the current version of the WebLogic Server. DDConverter is invoked with the command:

```
java weblogic.ejb20.utils.DDConverter [options] -d destDir  file1
[file2...]
```

In this command, the file is an EJB 1.0 deployment descriptor or a JAR file containing EJB 1.1 deployment descriptors. Some of the DDConverter options are:

- ***-d destDir:*** Directory for the output deployment descriptor.
- ***-EJBVer output EJB version:*** Specifies the output EJB version; default is 2.0.

Before you can deploy an EJB, it has to be packaged in a JAR or an EAR file.

EJB JAR File

The structure of an EJB JAR file consists of the EJB classes and interfaces, and the META-INF directory containing the deployment descriptors. Create a directory, source/ejb20/basic/container-Managed, and a directory source/ejb20/basic/containerMana-ged/META-INF. Copy Account.java, AccountBean.java, AccountHome.java, and ProcessingErrorException.java from the /weblogic81/samples/server/examples/src/examples/ejb20/basic/containerManaged directory to the source/ejb20/basic/container-Managed directory. Copy ejb-jar.xml, weblogic-ejb-jar.xml, and weblogic-cmp-rdbms-jar.xml from the /weblogic81/samples/serv-er/examples/src/examples/ejb20/basic/containerManaged direc-tory to the source/ejb20/basic/containerManaged/META-INTdirectory.

Create a JAR file from the compiled EJB Java class files and the deployment descriptors with Apache Ant, a Java-based build tool. The Apache Ant tool requires a build file. Create a build.xml file (see Listing 1) with targets to compile the EJB source files and gen-erate a JAR file from the compiled class files.

FORMULA ONE
Enterprise Reporting Toolsets for Java Projects & Portals
ReportingEngines
JAVA REPORTING TOOLS FROM ACTUATE

TROUBLE SEEING BEA PORTAL DATA?

NEW!

BEA PORTAL REPORTING SOLUTIONS

POWERED BY THE FORMULA ONE E.REPORT ENGINE

100% Pure Java: API-driven toolsets for building queries and creating reports against BEA Portal Server events.

Visual Access to BT_EVENT: Point-and-click wizards streamline report creation against the complex structure of BEA Portal Event and Behavior Tracking Data.

Deliver Reports Through BEA Portal Server: Embed reports that inherit the server features (security, connection pooling, etc.) of your BEA platform. No external report server to set up or maintain.

Analyze Server Events: Determine user traffic patterns. Find the most and least popular items in your Portal applications. Calculate click-throughs on marketing campaigns. And more!

Flexible: Available as an integrated WebLogic Workshop Extension or as a set of JAR files for use with JBuilder, IntelliJ IDEA, and other Java development environments.

888-884-8665 • sales@reportingengines.com • http://www.reportingengines.com/products/beaportal/overview.jsp

Copy the build.xml file to the /source directory. Run the ejb-jar target in the build.xml. An EJB JAR file is generated in the source/dist directory. The EJB JAR may be compiled with the BEA WebLogic appc compiler. Compiling with the appc compiler is not required, but it is recommended by BEA WebLogic.

Appc

The appc compiler generates container classes from the EJB JAR file and validates the deployment descriptors. The advantage of compiling the EJB JAR class files with appc prior to deploying the EJB JAR file is that the errors, which might be generated when the EJB JAR is deployed on the server, are identified. To run the appc compiler weblogic.jar should be in the Classpath. Appc is invoked with the command:

```
java weblogic.appc [options] <jar file or directory>
```

Some of the appc options are:

- ***-output<file>:*** Specifies a output directory
- ***-keepgenerated:*** Keeps the generated .java files
- ***-compiler<java>:*** Sets the Java compiler to use

Deploying an EJB

A data source with a Java Naming and Directory Interface (JNDI) name is required to deploy an entity EJB JAR file. Creating a Connection Pool is explained in my previous article (***WLDJ***, Vol. 3, issue 1). Create a Tx Datasource with the JNDI name "examples-dataSource-demoPool". The JNDI name should be the same as that specified in the <data-source-name> element in the weblogic-cmp-rdbms-jar.xml deployment descriptor.

To deploy the entity EJB JAR file select the Deployments>EJB Modules node in the administration console. Click on the Deploy a new EJB Module link (see Figure 1).

A Deploy an EJB Module frame is displayed (see Figure 2). Select the upload your files(s) link in the Deploy an EJB Module frame. An Upload and Install an Application or Module frame will be displayed. Select an EJB JAR file to upload and click on the Upload button.

"The principal motivation underlying EJB architecture is separation of concerns"

The Deploy an EJB Module frame is displayed. Select the myserver link. A list of subdirectories in the myserver directory gets displayed. Click on the upload directory link. In the upload directory select the EJB JAR to deploy and click on the Target Module button (see Figure 3). A Select Targets for this EJB module frame is displayed (see Figure 4).

Select one or more target servers in the Select targets for this EJB module frame and click on the Continue button. In the frame displayed, in the Name field, specify a name to be used for the EJB module to be deployed (see Figure 5).

Click on the Deploy button to deploy the EJB JAR file. The EJB JAR is deployed on the server and an EJB node gets added to the EJB Modules node (see Figure 6).

The EJB application may also be deployed to the BEA WebLogic Server by copying the EJB JAR file to the applications directory in the domain the application is to be deployed.

BEA WebLogic provides some recommendations for deploying an EJB on the WebLogic server.

1. EJBs should be deployed as an EJB JAR in an enterprise archive application (EAR application) to facilitate application migration and modification.
2. EJBs that have a reference to other EJBs should be deployed in the same application; the enable-call-by-reference element in weblogic-ejb-jar.xml should be set to True for better performance.
3. EJBs deployed on a WebLogic server cluster should be deployed homogeneously to each managed server in the cluster. If an EJB is deployed on only a single server in a cluster compile the EJB with the appc compiler before deploying.

Conclusion

EJB application development consists of creating the EJB classes, compiling the EJB classes and creating a JAR file from them, and deploying the EJB JAR to the BEA WebLogic Server. EJB application performance is improved by following some of the recommendations in designing, generating and deploying an EJB JAR to the BEA WebLogic Server.

Resources

- *Programming WebLogic Enterprise JavaBeans:* http://edocs.bea.com/wls/docs8.1/ejb/index.html

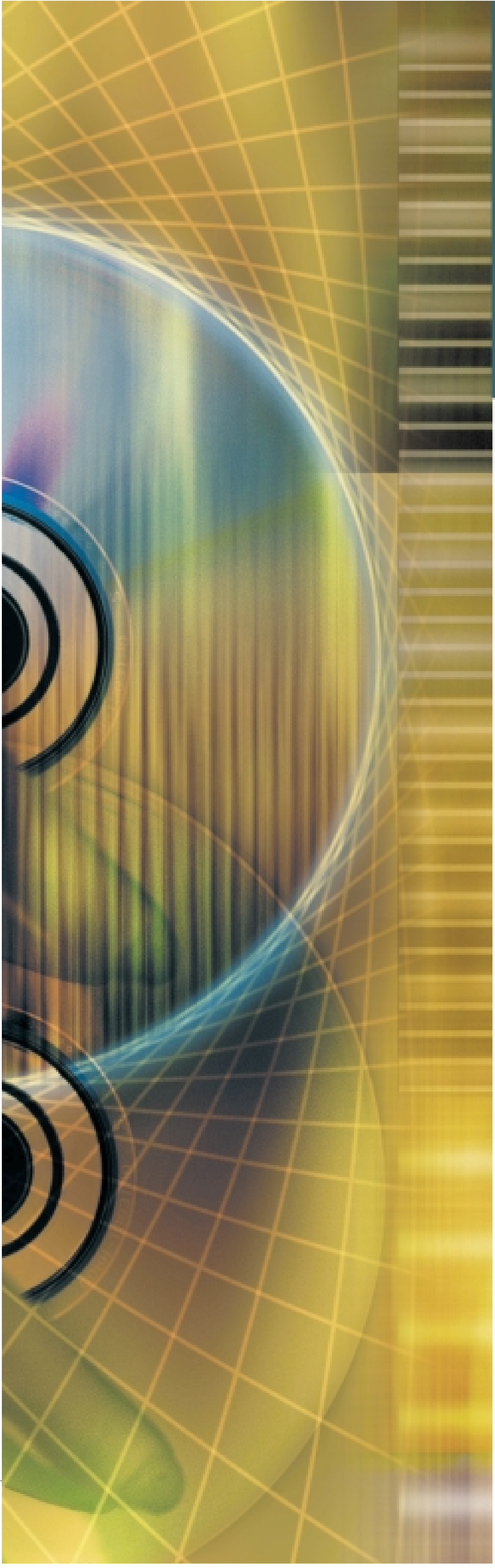
Listing 1: build.xml file

```
<project name="example-entity-ejb" default="all" basedir=".">

  <property name="source" value="."/>
   <property name="ejb" value="ejb20/basic/containrManaged"/>
  <property name="build" value="${source}/build"/>
   <property name="j2sdkee" value="c:/j2sdkee1.3.1"/>
  <property name="dist" value="${source}/dist"/>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
    <mkdir dir="${build}"/>
    <mkdir dir="${build}/META-INF"/>
    <mkdir dir="${dist}"/>
  </target>

  <target name="ejb">
    <javac srcdir="${source}" classpath="${j2sdkee}/lib/j2ee.jar" dest-
dir="${build}"
       includes="${ejb}/*.java"/>
    <copy  todir="${build}/META-INF">
    <fileset   dir="META-INF" includes="*.xml" />
    </copy>
  </target>
  <target name="ejb-jar" depends="ejb">
    <jar jarfile="${dist}/entityejb.jar" includes="META-INF/*.xml,
${ejb}/*.class"
      basedir="${build}"/>
  </target>
</project>
```



WHY INSTANCE-LEVEL PERFORMANCE DATA IS CRITICAL FOR J2EE APPLICATION MANAGEMENT

Application management is a relatively new software discipline, but has grown in significance as organizations increasingly rely on J2EE and legacy transactional middleware environments to support web-enabled enterprise applications. This technical brief examines the importance of monitoring application performance and behavior at an instance level rather than relying on aggregated data, and why it is necessary for developers, testers, and data center managers to have clear visibility within instance-layer functionality in order to successfully manage and monitor application middleware.

The Demand Burden of Application Software

One way to think of an application and its surrounding infrastructure is in the context of supply and demand. Application servers, with access to CPU, runtime memory, data resource pools, network bandwidth, and disk, supply the resources necessary to run applications. Conversely, applications represent the demand side of the equation: as software executes it generates demand on server resources by allocating objects, establishing connections, executing threads, sending messages, and other functions that impact server resources. As user activity increases, the application places increased demand on server resources.

Ideally, 3 conditions have been met for all application servers running in production to provide optimal supply: 1) servers have been properly tuned for the type of load or function, 2) infrastructure has been sized for the amount of transactions expected at peak load intervals, and 3) the application software is bug-free. While IT departments do a very good job at coming close, attaining 100% success in all 3 categories is very difficult to achieve and sustain. QA, load testing, capacity planning, redundant systems and other practices can be effective, but the complexity of all inter-dependent components working together under variable workloads makes it virtually impossible to assure completely error-free operations.

Instances, Requests, and Transactions

Cyanea uses the term *instance* to describe each request that is processed within a transactional middleware environment, for example a J2EE application server (in this paper the terms *instance* and *request* will be used interchangeably). Within J2EE, requests are generated by Java program components like servlets or Enterprise Java Beans (EJB's) whenever they are invoked by a user or another Java program.

J2EE requests (generally in the form of a URI or component string) encode and communicate Java instructions or program syntax to other programs or J2EE resources. This is an example of a J2EE URI generated by a servlet, which is part of a trading application (an application object is requesting a quote, supplied by another Java program, for a securities trade with a timeout of 100 milliseconds):

```
Trade/GetQuote?ttl=100
```

Let's see how requests can be monitored to detect defects, determine the causes of problems, and how request level information can be used to take corrective action when middleware issues arise.

–continued on the following page

Instance-Level Monitoring

Data center operations teams responsible for maintaining the reliability and performance of computing infrastructures traditionally run systems monitoring tools (sometimes referred to as *management frameworks*) to measure available resources, which generate alert messages when certain threshold conditions have been exceeded (such as high percentage CPU utilization or server unavailable event). Such frameworks are effective at managing infrastructure resources (the *supply side*) but are not able to detect and report instance-level application performance or stability problems on the demand side, which is often the most relevant place to look for causes of outages or other application performance problems.

Production J2EE application servers can scale to handle thousands of requests per second (total requests per second being measured as *throughput*). The amount of elapsed time needed to process a request is generally very small, and when measured on a well-tuned server may take only a few milliseconds or less. But sometimes requests take much longer, or even appear to "hang". And slow transactions – especially under production loads – are an unfortunately common issue. Cyanea/One monitors in-flight requests at the instance level, making it possible to detect the occurrence of any problematic request, and even determine which users are affected before the problem spreads to a larger group.

Other application management products that measure performance by calculating aggregated and averaged metrics simply cannot detect or display instance level performance anomalies. Such products are only able to detect issues after they have accumulated to the point where the average is affected. As a result, specific or intermittent problems will go undetected, and error conditions will exacerbate until they eventually manifest at a macro level, thus affecting potentially large numbers of users. Cyanea/One was developed for the express purpose of intelligently capturing all instance-level requests in every monitored JVM, and proactively gather performance data to alert IT management about problems before they spread.

Not only does Cyanea/One "see" every request, it provides elegant problem determination functions that enable users to drill down on problematic requests to gather important details about the problem, e.g. determine if a runaway looping condition is active, or an EBJ method is idle waiting for a return from a JDBC call. Capturing diagnostic information related to a request, such as session details, stack traces, thread dumps and other data, give support personnel the information they need to quickly isolate problems and figure out root causes – all from the perspective of the application.

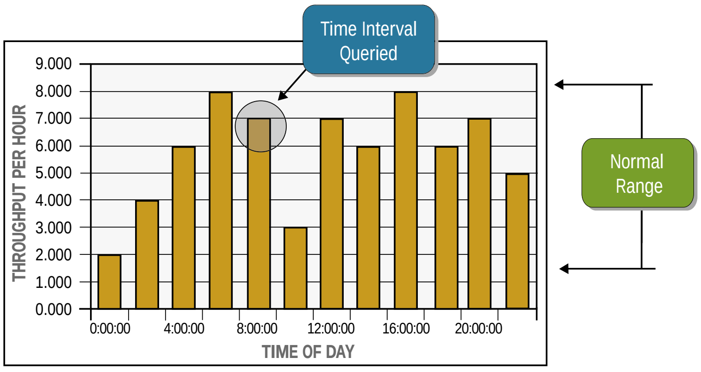
Example Problem Scenario

This is an example of how a slow transaction was isolated to a particular EJB method parameter.

Similar to In-Flight Request Search and Tracing functions, PAR makes use of instance-level requests, and presents an intuitive drill-down user interface for localizing granular sources of conditions that contribute to poor performance. This scenario steps through an example of how PAR is used to analyze and decompose J2EE requests in order to pinpoint a performance problem that does not otherwise show up within aggregated performance data, but repeatedly affects users under certain intermittent conditions.

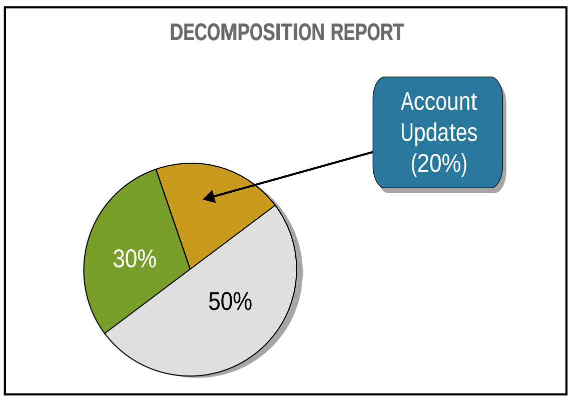
The main data center for a large bank recently received a number of support calls from customers complaining that customer account updates sometimes take a long time or even timing out, while other updates seem to work fine. A check of system resources did not reveal any problem conditions, so the issue was escalated to Level II software support. The Level-II J2EE support analysts decided to take a look at the application environment to determine if they can find anything in the performance data that could provide clues to the intermittent problem. The following steps were taken to produce gather and report performance data.

1. A *Request Transaction Analysis* report was produced based on all requests executed on the server group that running the customer account application. The output of this report (below) shows the aggregate throughput by hour of day, which does not reveal any abnormalities.



2. Analyst chose a bar from a time segment during which some of the complaints were received. Clicking on the bar decomposes the requests into the various types that were processed on the server.

3. The Decomposition Report shows that Account Update requests accounted for about 20% of the total as shown in the chart. So far every indicator appears to be within normal range.



4. Drilling down on the Account Update section of the chart reveals a list of all requests that were captured during the interval. The analyst then sorts the requests by response time, and immediately observes that both response time and CPU time for a small subset of the requests was many times greater than the rest of the transactions that appeared to complete within a nominal range.

5. Next the analyst sected one of the problematic request names to further drill down to view a Method and Component view of this particular transaction. In this view it appears that the account update for this particular user is taking a considerable amount of elapsed time as well as being unusually CPU intensive, as shown in the **bold red** values below

Nesting Depth	Event Type	Event Data	Elapsed time (ms)	CPU time (ms)	Elapsed Time (ms)	CPU time (ms)
6	Method Entry	cust/EJSJDBCPersisterCMPHolding Bean._update.eastCust	7.40	10.5	.01	.10
7	JDBC Entry	**Data Source Name: Customer 2A** **SQL Statement:** UPDATE customer RegistryBean SET account.region = east WHERE userID = ?	**8000**	**600.97**	1006	4.06

6. The above step is repeated for another problematic request, and displays a similar, consistent set of values for requests of this type.

7. Next the analyst selected a request that takes much less time for comparison, and notes there are certain differences between the two requests, as shown in the **bold blue** values below.

Nesting g Depth	Event Type	Event Data	Elapsed time (ms)	CPU time (ms)	Elapsed Time (ms)	CPU time (ms)
6	Method Entry	cust/EJSJDBCPersisterCMPHolding Bean._update.westCust	7.22	12.5	01	.10
7	JDBC Entry	**Data Source Name: Customer 3A** **SQL Statement:** UPDATE customer RegistryBean SET account.region = west WHERE userID = ?	**82**	**4.97**	13	.01

8. At this point the problem has been localized to a specific method invocation that results in a SQL update statement. Yet the question remains – why does the UPDATE statement take so much longer for requests for customers in the eastern region, while all the others customer regions perform normally?

9. At this point Level-II has localized the problem to a specific method invocation that only affects a specific customer group. Upon further investigation by the database administration group, it was determined that the table structure for customers in the database instance was not indexed properly, as well as being undersized. Once the administrative corrections and table optimizations were completed, the account updates for customers in the Eastern region resumed execution without noticeable delay.

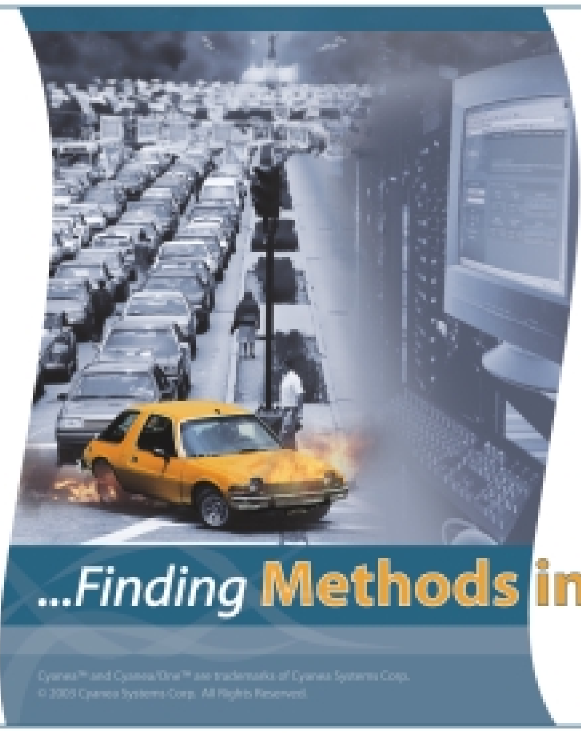
Conclusion

As illustrated in the above example, capturing transaction-level data within a production application server environment is critical when tracking down root causes of typical performance problems. Since many performance issues are isolated in nature, they may go undetected when simply measuring aggregated response time, and are difficult to debug without seeing actual requests and drilling down to an appropriate level. Cyanea/One is unique in its ability to help IT personnel rapidly detect and resolve performance issues before they spread, and contributes directly to cost savings and customer satisfaction by keeping middleware environments healthy at all times.

PAID SUBMISSION BY CYANEA SYSTEMS CORP.

Cyanea™

www.cyanea.com



Cyanea®

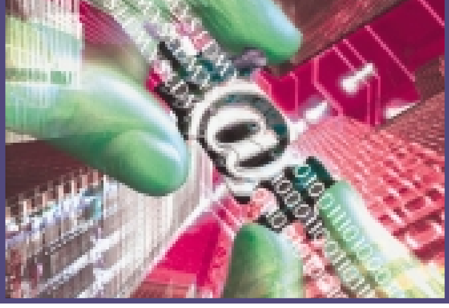
What's Obstructing Your Application Flow?

With Cyanea/One™, your company will gain valuable insight into its J2EE applications. With a few mouse clicks, you can quickly drill down into your applications, all the way down to the method level. Isolate, diagnose and proactively resolve bottlenecks and resource consumption issues. Accomplish all these with-out touching your code or even requiring knowledge of your application.

...Finding Methods in the Madness.

Cyanea™ and Cyanea/One™ are trademarks of Cyanea Systems Corp.
© 2005 Cyanea Systems Corp. All Rights Reserved.

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



'HTTP Session Replication Failure' Issues

MOVING AWAY FROM THE PRIMARY THOUGHT

BY **VIMALA RANGANATHAN**

AUTHOR BIO

Vimala Ranganathan is a backline developer relations engineer with BEA Systems. She specializes in troubleshooting and solving complex customer issues with their mission-critical applications on BEA products. Vimala holds a bachelor's degree in computer science.

CONTACT...

rvimala@bea.com

Sometimes, HTTP session states are not replicated from the primary server to the secondary server.

Symptoms

1. The application using HTTP session does not function as designed and you see a loss of session data.
2. You might be asked to re-log into the application even when the session has still not timed out.
3. You see errors and warnings related to HTTP session failures in the server's log file.
4. The request is not failed over to another server properly.

There are several reasons as to why session replication fails. We will look at ways to diagnose the issue, possible reasons for it, and the ways to address them.

Types of HTTP Session Replication

There are five different implementations of session persistence:

- ***Memory (single-server, non-replicated):*** When you use memory-based storage, all session information is stored in memory and is lost when you stop and restart WebLogic Server.
- ***File system persistence:*** Session information is stored in a file under the PersistentStoreDir specified.
- ***JDBC persistence:*** Session information is stored in a database table.
- ***Cookie-based session persistence:*** Session information is stored in a cookie.
- ***In-memory replication (across a cluster):*** Session data is copied from one server instance to another into memory.

Diagnosing the Issue

Consider a scenario with two servers (MyServer-1, MyServer-2) in a cluster. When you enable the debug flags (see the "Enable the Debug Flags to Track Session Replication Failures"section), you will see the messages below when a request from a client is sent the very first time.

On MyServer-1:

```
<Oct 9, 2003 12:38:21 PM PDT> <Debug> <Cluster>
<000000> <Creating primary 5165892837402719733>
<Oct 9, 2003 12:38:21 PM PDT> <Debug> <Cluster>
<000000> <Created secondary for 5165892837402719733 on
-7957889153726652135S: 192.168.11.112: [9001,9001, -1,
-1,9001, -1, -1]: mydomain: MyServer-2>
```

This logging message means that the primary server is MyServer-1 and a secondary has been created on MyServer-2. A message like the one shown below will be logged on MyServer-2 to confirm that.

```
ExecuteThread: '1' for queue: 'Replication'> <kernel
identity> <> <000000> <Creating secondary
5165892837402719733>
####<Oct 9, 2003 12:38:21 PM PDT> <Debug> <Cluster>
<rvimala-c840> <MyServer-2> <ExecuteThread: '1' for
queue: 'Replication'> <kernel identity> <> <000000>
<Updated local secondary of 5165892837402719733>
```

If you check the JSESSIONID it looks like:

```
JSESSIONID=1E9Xwn7nLYfOsc1obgRZIwW5s72an7HPPvSD7iaWHMXz
pHga5cQj!-1587343083!-1587348922
```

JSESSIONID is the default name of the cookie,

Now
More
Than
Ever
Before...

JAVA DEVELOPER'S JOURNAL
JDJ

Means
Business

EXCLUSIVE: WHERE IS TECH GOING IN 2004?
JAVA DEVELOPER'S JOURNAL
JDJ
JANUARY 2004 VOLUME:9 ISSUE:1
SPECIAL REPORT: SCOTT McNEALY 2004 PREDICTIONS
PLUS...
Viewpoint: Deutsch's Fallacies, 10 Years After
Feature: Multiplatform Frames
Data Integration: Which EII Solution Is Right for You?
EDGE 2004 EAST
BOSTON, MA
TAKE A RIDE WITH THE CEO OF BEA
THE WORLD'S LEADING i-TECHNOLOGY MAGAZINE WWW.JAVADEVELOPERSJOURNAL.COM

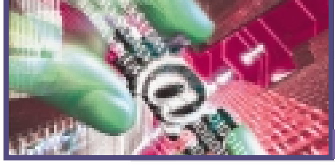
Subscribe Now!
$69^99
1 YEAR/
12 ISSUES

www.SYS-CON.com/JDJ
or call
1-888-303-5282

The World's Leading *i*-Technology Magazine Just Got Better!
Now with expanded business coverage, *JDJ* is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of IT professionals.

The Most Popular
i-Technology Magazine
in the World !

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



which could be changed to anything in weblogic.xml. The format of JSESSIONID is

```
SessionId!PrimaryServer JVM Hash!SecondaryServer JVMHash
```

Every time data is changed (either set/get or removed) in the session you'll see the logging message.

On MyServer-1:

```
<Oct 9, 2003 12:38:21 PM PDT> <Debug> <Cluster> <000000> <Updated remote
secondary for 5165892837402719733>
```

On MyServer-2:

```
####<Oct 9, 2003 12:38:21 PM PDT> <Debug> <Cluster> <rvimala-c840>
<MyServer-2> <ExecuteThread: '1' for queue: 'Replication'> <kernel identi-
ty> <> <000000> <Updated local secondary of 5165892837402719733>
```

If for any reason session replication fails, you will see the message below in the MyServer-1 log:

```
<Nov 6, 2003 12:59:12 PM EST> <Debug> <Cluster> <000000> <Unable to create
secondary for -5165892837402719733>
<Nov 6, 2003 12:59:12 PM EST> <Debug> <Cluster> <000000> <Error creating
secondary 5165892837402719733 on -7957889153726652135S:
192.168.11.112:[9001,9001,-1,-1,9001,-1,-1]:mydomain:MyServer-2>
```

And, the JSESSIONID would appear as

```
JSESSIONID=1E9Xwn7nLYfOsc1obgRZIwW5s72an7HPPvSD7iaWHMXzpHga5cQj!-
1587343083!NONE
```

The secondary server hash would become NONE

Enable the Debug Flags to Track Session Replication Failures

You can enable the flags DebugCluster, DebugClusterAnnouncements, DebugFailOver, DebugReplication, and DebugReplicationDetails.

To Enable

Use the weblogic.Admin command-line utility to dynamically turn the debug options on and off. For example, to turn on DebugCluster on all administration instances of ServerDebug MBean (i.e., Admin Server or al Managed Server):

```
java weblogic.Admin -url t3://localhost:7001 -username system -password
weblogic SET -type ServerDebug -property DebugCluster true
```

Or, edit the config.xml and the MBean element in the <ServerDebug/> stanza for each server that you want to debug and set it to a value of "true" to enable or "false" to disable. Then you must restart the Admin Server. Managed Servers will reconnect to the Admin Server and the debug flags will then dynamically take effect. For example:

```
<ServerDebug DebugCluster="true" Name="myserver"/>
```

At the end, with all the flags set, in your config.xml the ServerDebug tag would look like:

```
 <ServerDebug ClassFinder="true" DebugCluster="true"
DebugClusterAnnouncements="true" DebugFailOver="true"
DebugReplication="true" DebugReplicationDetails="true" Name="MyServer1"/>
```

Make sure the stdOutSeverity level of the server is INFO and StdoutDebugEnabled is set to "true". The debug information will be logged into the server log as well as to the standard out.

Checklist for Each Session Persistence Type

Memory (Single Server, Nonreplicated)

1. When you use memory-based storage, all session information is stored in memory and is lost when you stop and restart BEA WebLogic Server.
2. Make sure you have allocated sufficient heap size when running WebLogic Server; otherwise, your server may run out of memory under heavy load.
3. Not a recommended type for Cluster configuration (because the data is kept in heap and is not available to any other server).

File System Persistence

1. Verify that the directory where BEA WebLogic Server stores the sessions is correctly specified in weblogic.xml. You must also create this directory yourself and make sure appropriate access privileges have been assigned to the directory.
2. Make sure you have enough disk space.

JDBC Persistence

Make sure the connection pool that connects to the database has read/write access for the database table used.

Cookie-Based Persistence

1. Make sure you have not stored anything other than java.lang.String in the HTTP session.
2. Do not flush the HTTP response object in your application code.
3. Make sure that the content length of the response exceeds the buffer size set (default is 8192 bytes).
4. Make sure that cookies are enabled in the browser.
5. Make sure that you do not use commas (,) in a string when using cookie-based session persistence.

In-Memory Replication

1. Make sure that the BEA WebLogic Server is accessed only via a proxy server or a hardware load balancer
2. The hardware load balancer should support a compatible passive or active cookie persistence mechanism, and SSL persistence.
3. Recommended type in a cluster.

Validate the weblogic.xml Entries

Make sure weblogic.xml has all the parameters that need to be set for each Session Replication type. For example, when using in-memory replication the sample weblogic.xml would look like:

```
<session-descriptor>
  <session-param>
     <param-name>
       PersistentStoreType
     </param-name>
     <param-value>
       replicated
     </param-value>
```

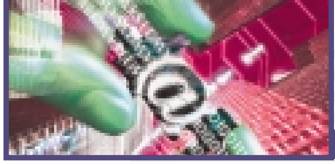
A new tool for MX professional developers and designers...

ADVERTISE
Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE
www.sys-con.com/
mx/subscription.cfm
1(888) 303-5282

SYS-CON MEDIA

MX
developer's journal



```
  </session-param>
</session-descriptor>
```

Note: The debug files are subjected to change with each version of BEA WebLogic Server.

Session Data Must Be Serializable

To support in-memory replication of HTTP session states, all servlet and JSP session data must be serializable or else session replication would fail.

When debugging is enabled, BEA WebLogic Server would output the warning messages below by indicating that the session is not replicated. After this, session replication would stop.

Debug Message:

```
<Oct 8, 2003 2:10:45 PM PDT> <Error> <Cluster> <000126> <All session
objects should be serializable to replicate. Please check the objects in
your session. Failed to replicate non-serializable object>
```

On the subsequent request the JSESSIONID will be missing the secondary server and will be marked NONE, e.g.:

```
JSESSIONID=1E9Xwn7nLYfOsc1obgRZIwW5s72an7HPPvSD7iaWHMXzpHga5cQj!-
1587343083!NONE
```

Solution: Find out the page the error is thrown from and make sure that all the data put into the session is serializable.

Check for Network/Multicast Issues

Make sure that the network is fine and there are no multicast issues. Do the multicast test to make sure that the multicast IP is working fine.

Syntax

```
java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout]
[-s send]
```

(http://edocs.bea.com/wls/docs81/adminref/utils.html#1199798)

Validate Cluster Configuration

The primary and secondary servers are selected from the cluster list. In a cluster of two servers, if the cluster doesn't have all the servers, then a secondary might not be chosen, resulting in the session data not being replicated.

To verify, execute the following commands:

1. Make sure weblogic.jar is in the classpath.
2. To get all the servers in a cluster:

```
java weblogic.Admin -username weblogic -password weblogic -url http://one-
ofthemanagedserverurlinthecluster:7001/ GET -type ClusterRuntime –pretty
```

This will list all the servers in a cluster. The URL could be changed to every server in the cluster to make sure they all have the same entries.

Application Code Diagnostics

Make sure you use only setAttribute/removeAttribute methods of the HTTP session in your application code to update the HTTP session. If you use other set methods to change objects within a session, BEA WebLogic Server does not replicate those changes.

Please do not use the methods putValue and removeValue of the HTTP session as they are deprecated and there could be issues with session data replication when using such methods in your application. Instead, use only the setAttribute/removeAttribute methods of the HTTP session.

Cookies vs URL Rewriting

In some situations, a browser or wireless device may not accept cookies, which makes session tracking with cookies impossible. URL rewriting is a solution to this situation that can be substituted automatically when WebLogic Server detects that the browser does not accept cookies. Enable URL rewriting in BEA WebLogic Server by setting the URLRewritingEnabled attribute in the WebLogic-specific deployment descriptor, weblogic.xml, under the <session-param> element. The default value for this attribute is true.

Performance Issues

- ***Consider serialization overhead:*** Serializing session data introduces some overhead for replicating the session state. The overhead increases as the size of the serialized objects grows. If you plan to create very large objects in the session, test the performance of your servlet to ensure that performance is acceptable.
- ***Control frame access to session data:*** If you are designing a Web application that utilizes multiple frames, keep in mind that there is no synchronization of requests made by frames in a given frameset. For example, it is possible for multiple frames in a frameset to create multiple sessions on behalf of the client application, even though the client should logically create only a single session.

  To avoid unexpected application behavior, carefully plan how you access session data with frames. You can apply one of the following general rules to avoid common problems:
  – In a given frameset, ensure that only one frame creates and modifies session data.
  – Always create the session in a frame of the first frameset your application uses (for example, create the session in the first HTML page that is visited). After the session has been created, access the session data only in framesets other than the first frameset.
- ***Storing larger amounts of data in the session:*** JDBC persistence and File persistence won't be faster as the session data has to be stored and retrieved from an external resource. There is also a performance overhead because of JDBC access for each session update. If you want to store large objects in the session, then JDBC or File persistence should be considered.
- ***Storing small amount of data in the session:*** Cookie-based session persistence is most useful when you do not need to store large amounts of data in the session. Cookie-based session persistence can make managing your BEA WebLogic Server installation easier because clustering failover logic is not required.

Further Information

For additional information go to http://support.bea.com for some published solutions on session replication failures. You can also query ASK BEA at http://websupport.beasys.com/index.jsp.

If none of these help you towards a solution or an identifier in your application, then contact BEA Customer Support for further diagnosis. You can open a case with a valid support contract by logging in at http://support.bea.com/login.jsp

INFORMATION
STORAGE+
SECURITY
JOURNAL
ISSJ PREVIEWING MAY 11, 2004 NETWORLD+ INTEROP IN LAS VEGAS
Microsoft
RSA Security
SecurID for
Windows
SAN
NAS
Emerging technology trends
and market maneuvers
PLUS:
Secure Service-Oriented B2B Architectures
Rebuilding the Foundation of Enterprise Security
Disaster Recovery with an iSCSI SAN
IP STORAGE • SAN NAS • DISASTER RECOVERY • ENTERPRISE SECURITY • INFOSECURITY

Modernizing Legacy Systems, PART 3

WORKFLOW AND SERVICES-ORIENTED ARCHITECTURE

This is the last installment in my series of articles on modernizing legacy systems with the BEA WebLogic Platform (**WLDJ**, Vol. 3, issues 2–3). Part 1 introduced a high-level "modernization process," with a "recipe" towards modernizing legacy systems. In Part 2, I concentrated on integrating the legacy system with the WebLogic Platform. I also introduced the concept of a layered architecture and some fundamental J2EE patterns .

BY **ANWAR LUDIN**

AUTHOR BIO

Anwar Ludin specializes in service-oriented architectures for the financial sector. He currently works as an independent consultant for financial institutions in Switzerland, where he helps design J2EE architectures based on the BEA WebLogic Platform.

CONTACT...

anwar.ludin@agilethinker.com.

In many ways Parts 1 and 2 represented the fundamental "building blocks" in modernization. This month I'll concentrate on building the business process, or domain model layer. I will also introduce the concept of application services, which is a way of managing the workflow or use cases of the modernized platform. Finally, I'll look at how to build a service-oriented architecture with BEA WebLogic Workshop. As I mentioned in the first installment, one way to summarize "modernization" is by saying that it's a way of incrementally moving business logic from a legacy system to WebLogic.

Business Objects and Domain Models

As I discussed in Part 1, one of the tasks in the modernization process is to "prioritize use cases" in order to port them to the BEA WebLogic Platform. At the time, I didn't really get into the details of how to actually do the porting. Let us now get into those details by studying the business layer of our architecture. The advantage of using an object-oriented language such as Java over a procedural language such as COBOL is that you can actually build a rich domain model. Basically, the domain model will be a layer of interconnected objects with associations and inheritance hierarchies. The domain model will usually be implemented using business objects with fine-grained interactions that have state and behavior. A crucial step in the modernization effort is therefore to design the domain model so that it reflects as closely as possible the business entities used by the legacy system. In other words, the domain model is an object-oriented implementation of the conceptual model we have devised by studying the legacy system. In Figure 1, for example, for a banking system, entities such as accounts, customers, credit cards, ranking strategies, and so on will be mapped into business objects with relationships, inheritance hierarchies, and behavior.

Business objects are nothing more than "plain old Java objects," or POJOs, so they can be tested in an independent way and even reused. It should be noted here that I am making a distinction between business objects, which are POJOs, and entity beans. For me, an entity bean is a persistence mechanism but is not adapted for building a rich domain model. You might not actually agree with my point of view and consider that the domain model can be designed with entity beans, but I personally have found that approach very cumbersome. However, nothing stops you from "persisting" the business objects by using CMP entity beans or, as shown in Part 2, by using DAOs to the legacy system. Now let's look at the concept of "application services," which implements the layer on top of our domain model.

Application Services

You might decide to implement the use cases to be modernized (see Part 1) by using Session Facades. In this case, the Session Facades would directly leverage business objects from the domain model and also contain some logic specific to the use case. In such a scenario, you would have as many Session Facades as modernized use cases. However, this approach has the following disadvantages:

- We are using your Session Facades for the wrong intent. Session Facades are only meant to be used for exposing business-tier components to remote clients as coarse-grained services.
- Our Session Facades become bloated with business logic.
- There is a risk of reproducing use case business logic between Session Facades.

The solution to the previous problems is to use "application services". Application services provide a way of encapsulating use case–specific logic outside of individual business or domain objects. In other words, every use case to be modernized should be implemented by one or several application services that in turn update the domain objects. Another way of seeing things is that domain objects provide the static relationships in the business domain and application services provide the dynamic interactions between the domain objects.

Finally, Session Facades can leverage application services in order to communicate with remote clients. Figure 2 illustrates the relationship between session facades, application services, and business objects in our banking example.

One thing I absolutely like about J2EE is that it promotes component-based design. BEA WebLogic Workshop helps me design a service-oriented architecture by building Web services from my J2EE components but at the same time hiding the underlying complexities of J2EE development. Very powerful indeed. I'll start by defining what I mean by "service-oriented architecture." From my point of view, a service-oriented architecture is a way of building enterprise software so that its functionality can be accessed by "external" applications without relying on "internal" artifacts such as client JARs containing component interfaces. Basically, according to my definition, the service-oriented architecture simply adds a "Web services layer" to our layered architecture, which "hides" the underlying component platform. In that sense, Web services still rely on the underlying layers, such as application services, and the business-specific layer in order to fulfill their job. As you might recall, the business-specific layer contains all the reusable components. So what are the advantages of building such an architecture?

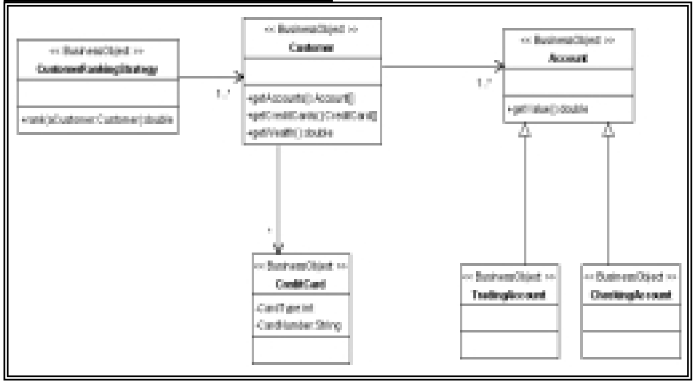
Basically, by adding the Web services layer to the modernized platform, we are promoting loose coupling between systems, which is good because we can evolve the platform without any impact on the clients leveraging our services. That is, of course, as long as our Web services' interfaces don't change. As a result, the modernized system is now seen as a set of services accessible from any internal IT system as well as external business partners. Figure 3 illustrates the overall architecture with the additional Web services layer.

Building Our Web Services with WebLogic WorkShop

The process of building the Web services layer of the modernized platform can be broken down roughly into the following steps:

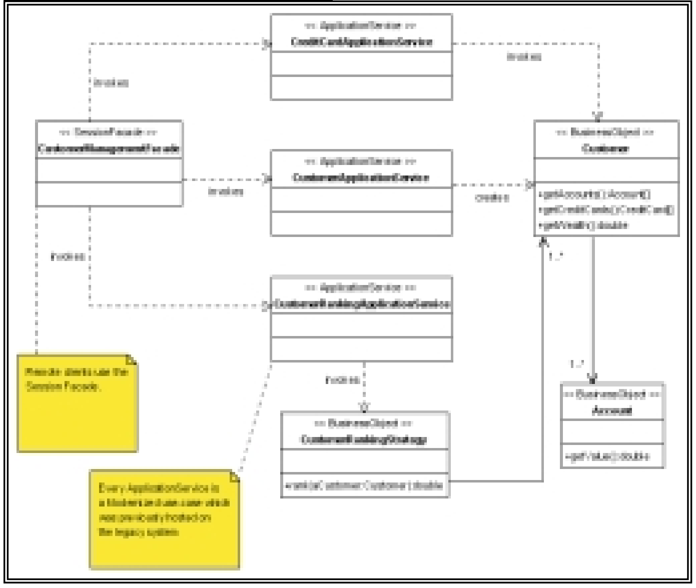
- Import into BEA WebLogic Workshop the required components from the business layer. If your Web services are hosted in the same WebLogic Server containing your business domain modules, then you will simply have to import the application services modules. However, if you decide to build your Web services as remote clients of the business tier, then you will have to import your Session Facade and transfer object modules. You don't need to import any components from the integration layer because they are "hidden" away from you by the business layer. Every layer in our architecture hides the underlying layer and transfer objects are used to communicate between layers.
- Design the Web service interface and its workflow. Will it be synchronous or asynchronous? Will it leverage another service? You will most probably design this interface with a business analyst. You can use activity and use-case diagrams during the design process. Basically, the Web service will expose to the "external world" a specific business process relying on the application services of the modernized platform.
- Code the service logic using BEA WebLogic Workshop by leveraging the components from the business layer. According to the architecture I have presented so far, the logic provided by the Web service should leverage as much as possible the underlying application services. In that sense, by using WebLogic Workshop you are actually building your Web service by "assembling" or leveraging application services from the business layer. In other

FIGURE 1



Business objects hierarchies

FIGURE 2



Application services layer service-oriented architecture words, WebLogic Workshop provides the "glue" to build Web services out of your components. I have found the approach to be extremely powerful.

It's that simple. BEA WebLogic Workshop really takes care of a lot of details specific to J2EE for you. Actually, one way of realizing how WebLogic Workshop simplifies your life is by trying to build the same service "manually," using Apache Axis for example.

Deploying Our Web Services with WebLogic Workshop

Deploying your Web services mainly consists of packaging them with the application services modules or Session Facade client JARs in an application archive. This is actually taken care of for you by BEA WebLogic Workshop, which will generate the EAR archives ready to be deployed. You simply have to select "Build EAR" from the "Build" menu in WebLogic Workshop. The result will be an EAR archive ready to be deployed to a production server. As shown in Figure 2, you actually have several possibilities for the runtime deployment of your services. For example, you might decide to package the entire modernized application with the Web services layer as a unique application archive or for security reasons separate the Web services layer from the rest of the modernized application by deploying them in different runtime nodes. In that case you would build two applications EARs, one for the modernized platform business and integration layers and another one for the Web services layer.

Conclusion

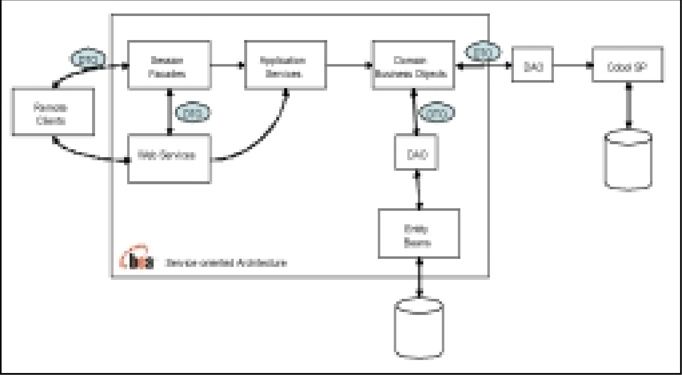
This was the last part in my series of articles about "modernizing" legacy systems with the BEA WebLogic Platform. I tried to provide a recipe-based approach for the different steps involved in the modernization effort. One of the things I also tried to emphasize is that any modernization effort should try to build a better, more robust platform on top of WebLogic and the J2EE specification. The modernized platform should provide additional, or value-added, services in order to justify the effort in the first place. Some of those services, such as security, container-managed persistence, and so on, are provided by the BEA WebLogic Platform. Other services have to be designed by us. However, building the "right" application architecture is extremely important.

I have tried to introduce several patterns that you can reuse in your own design. We have seen that we should build the modernized platform in layers, using the layered architecture approach. Integrating both modernized and legacy systems should be done by using Data Access Objects and Transfer Objects. The approach I have taken is "use-case driven" in the sense that we have tried to identify specific use cases of the legacy system and "re-hosted" them on the modernized platform. Those use cases are then mapped to one or several application services of the business layer. The modernized workflow results from the relationships and choreographies between application services.

I looked at the importance of building a rich domain model, and described the legacy system by designing relations and hierarchies between business objects. Besides giving us a better understanding of the business domain, the domain model provides us with an object-oriented view of the legacy system that was previously built by using a procedural language such as COBOL. I mentioned the advantages of building a services-oriented architecture and illustrated how easily this could be done using BEA WebLogic Workshop.

We used several patterns in the modernization effort (see Table 1), which are thoroughly documented. You can find out more about them by consulting the *Core J2EE Patterns, 2nd Edition* and *Patterns of Enterprise Application Architecture* books. Finally, I hope my articles provided some insight into "modernizing" a legacy system by using a rich, multilayered, service-oriented target architecture based entirely on BEA WebLogic.

FIGURE 3



Modernized service-oriented architecture

TABLE 1

CORE J2EE PATTERNS	PATTERNS OF EAA	PURPOSE
Transfer Object	Data Transfer Object	Carry data in bulk between tiers to minimize costly remote calls
Data Access Object	Data Mapper	Abstract the domain model from its persistence mechanism
Business Objects	Domain model	An object model of a specific domain, such as banking, that has behavior and data
Application Services	Service layer	Implement use cases acting on multiple business objects.
Session Facade	Remote facade	Expose business components and services as a coarse-grained facade to remote clients in order to improve efficiency over the network

Enterprise application architecture patterns

LOOK FOR YOUR FREE...

›Linux ›Java ›Web Services ›.NET ›XML ›Wireless ›Storage ›Security
SYS-CON MEDIA
The Premier Resource for Today's Corporate & IT Decision Makers
VOL 1 ISSUE 1 SPRING 2004
IT solutions ›GUIDE
WWW.SYS-CON.COM/IT
» Delivering Software as Service
» Leveraging Linux/Open Source
» Moving to a Service-Oriented Architecture
» Desktop Software: Migrating from Server to Client
» Using Developer Tools to Drive Cost Out of Software
» Application Integration
» Storage & Security
TECHNOLOGIES YOU NEED NOW!
How to Manage Your Ideas Using Today's i-Technologies
Reaching 135,000 Corporate and IT Decision Makers
$5.99US $7.99CAN

Coming this **SPRING!**

Application Management with WebLogic Server for Developers, PART 5

WRITING CUSTOM JAVA APPLICATIONS

BY **ROBERT PATRICK & VADIM ROSENBERG**

AUTHOR BIOS...

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

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

CONTACT...

rpatrick@bea.com
vadimr@bea.com

This article is the fifth in a series of articles on BEA WebLogic Server administration and management for developers (see **WLDJ**, Vol 2, issues 10–12; Vol. 3, issue 2).

We have focused on WebLogic Server administration concepts and terminology, the graphical tools for packaging an application and setting up and configuring a WebLogic Server domain; the application deployment, runtime management, and monitoring facilities available that did not require knowledge of JMX; basic concepts and terminology of JMX and the BEA WebLogic Server 8.1 JMX infrastructure; as well as how to use JMX-specific tools that come with WebLogic Server 8.1. In our last article, we showed you the basics of how to write custom Java applications that use JMX to configure, administer, and manage WebLogic Server 8.1–based applications.

This month, we'll continue our discussion of JMX programming by showing you how to use the notification facilities to create notification listeners, monitors, and timers.

JMX Notification

In addition to the ability to manipulate MBean attributes and operations, JMX also provides notifications when changes to MBean state are made. A JMX MBeanServer allows a management program to register its interest in these notifications on individual MBeans or on the MBeanServer itself. To register interest in JMX notifications, a program simply needs to create an object that implements the NotificationListener interface and register this object with the MBean or the MBeanServer. The NotificationListener interface has only one method, handleNotification(), which is the method that the MBean invokes to notify the listener:

```
public interface NotificationListener
extends java.util.EventListener
{
  void handleNotification(Notification n, Object hand-
back);
}
```

The Notification class contains information about the circumstances surrounding the event. The optional handback argument simply hands back whatever object was passed in when registering the listener. This provides a mechanism by which the listener can associate information about the MBean that's producing the notification that gets passed back to the listener without modification during the notification callback.

JMX 1.0 defines five types of notification objects that contain information specific to the type of notification:

- ***MBeanServerNotification:*** Used by the MBeanServer to notify listeners of MBean registration and deregistration
- ***AttributeChangeNotification:*** Used by MBeans to notify listeners when an attribute value changes

FREE*CD! ($198.00 VALUE!)

Secrets of the ColdFusion Masters

Every ***CFDJ*** Article on One CD!

COLDFUSION Developer's Journal
PRESENTS
ColdFusion
Resource CD
Secrets of the ColdFusion Masters
$198 VALUE
Inside:
Every Issue of CFDJ Ever Published on One CD!

— The Complete Works —

CD is edited by ***CFDJ*** Editor-in-Chief Robert Diamond and organized into 23 chapters containing more than 450 exclusive ***CFDJ*** articles!

All in an easy-to-navigate HTML format! **BONUS: Full source code included!**

ORDER AT WWW.SYS-CON.COM/FREECD

**PLUS $9.95 SHIPPING AND PROCESSING (U.S. ONLY)*

SYS-CON MEDIA

Only from the World's Leading i-Technology Publisher

- ***MonitorNotification:*** Emitted by the monitoring services when a specific set of conditions is met
- ***RelationNotification:*** Emitted by the relationship service when a relation is added, updated, or removed
- ***TimerNotification:*** Emitted by the timer service when a timer goes off

In addition, BEA WebLogic Server 8.1 also defines the following notification types:

- ***WebLogicLogNotification:*** Emitted every time an entry is written to the WebLogic Server log file
- ***AttributeAddNotification:*** Emitted every time an element is added to an MBean attribute that is an array
- ***AttributeRemovedNotification:*** Emitted every time an element is removed from an MBean attribute that is an array

Listing 1 is a very simple example of a listener that does nothing more than print out the details of an AttributeChangeNotification.

There are two important points that we need to make here. First, you will notice that we have made the class serializable. The notification listener mechanism, like the other JMX interfaces, does not account for remote access. To be able to use our standalone client to register the listener, you need to make the listener serializable because the call to addNotificationListener() will need to pass a copy of the listener to the server. This means that the listener we just created will actually be instantiated on the server and only the server-side copy of the listener will actually receive the notifications. In fact, you'll need to make sure that the listener class is in the server's CLASSPATH for this to work. Fortunately, BEA WebLogic Server provides an extension to allow you to easily create and register a listener that runs in a remote process.

To enable remote listener notification, all you need to do is change the above class to implement the weblogic.management.RemoteNotificationListener interface instead of the javax.management.NotificationListener and java.io.Serializable interfaces. This eliminates the need to have the listener class in the server's CLASSPATH and allows the notifications to be sent back to your client-side listener.

Second, you have to filter the types of notifications you are interested in by checking the actual class of the notification object passed into the listener. While this is not a big deal in our little example, you can imagine that it gets more unwieldy as you increase the number of notification types that the listener needs to manage. Fortunately, JMX defines a filtering mechanism that allows the registering program to describe the types of events the listener wants to receive. All notification filters must implement the javax.management.NotificationFilter interface shown here:

```
public interface NotificationFilter extends
java.io.Serializable
{
    public boolean
isNotificationEnabled(Notification notification);
}
```

The isNotificationEnabled() method returns true if the notification should be delivered to the filter and false otherwise. Notice that the NotificationFilter interface extends java.io.Serializable. The filter, much like the NotificationListener described earlier, will be copied and instantiated on the server. In this case, that is precisely what we want since we want the server to pass only the events of interest back to the client. Now we can move the filtering logic out of the listener itself and into the filter (see Listing 2).

Of course you can do much more sophisticated things with filters but a full discussion is beyond the scope of this article. Please refer to the JMX 1.0 specification (http://jcp.org/aboutJava/communityprocess/final/jsr003/) and the BEA WebLogic Server documentation (http://edocs.bea.com/wls/docs81/jmx/notifications.html) for more information.

“In addition to the ability to manipulate MBean attributes and operations, JMX also provides notifications when changes to MBean state are made”

To register your notification listener with an MBean, you simply invoke one of the addNotificationListener() methods on the MBeanServer or, if you are using WebLogic Server's strongly typed MBean interface, the addNotificationListener() method defined by the javax.management.NotificationBroadcaster interface that every WebLogic Server MBean implements. The following code snippet from the downloadable example (available online at www.sys-con.com/weblogic/sourcec.cfm) demonstrates using the MBeanServer interface:

```
String serverName =
"mydomain:Name=myserver,Type=Server";
ObjectName serverMBeanName = new
ObjectName(serverName);
MyAttributeChangeListener me =
    new MyAttributeChangeListener(serverName);
MyAttributeChangeFilter filter = new
MyAttributeChangeFilter();
mbeanServer.addNotificationListener(serverMBeanNa
me, me, filter, null);
```

With JMX Notification, the registered listeners receive notification any time an event for which they have registered their interest occurs. If the management application really just wants to monitor the value of an MBean attribute, JMX provides another facility to accomplish this known as JMX Monitoring.

JMX Monitoring

The JMX specification requires that a JMX MBean server provide a monitoring service. JMX monitoring is surfaced through a set of MBeans known as Monitor MBeans. A management application configures these Monitor MBeans to periodically observe other MBeans and emit a JMX notification if a specific MBean attribute exceeds the configured threshold. The value of the attribute being observed, known as the derived gauge, can either be the exact value of the attribute or the difference between two consecutive observed values (if the attribute is numeric). The frequency with which the monitor samples the value of the observed attribute is called the granularity period.

Monitor MBeans come in three flavors:

- ***CounterMonitor:*** Observes integer-type attributes that behave like a counter in that the values are always greater than or equal to zero and the values are only incremented (though they can rollover)
- ***GaugeMonitor:*** Observes numeric

attributes that behave like a gauge in that the values can arbitrarily fluctuate
- ***StringMonitor:*** Observes string attributes

Whenever the value of the derived gauge exceeds the configured threshold, the Monitor MBean generates a JMX notification. These notifications are sent using an instance of the MonitorNotification class, which is a subclass of the Notification class discussed earlier. This class contains information such as the notification type, the observed MBean name, the observed attribute name, the derived gauge, and the threshold value that triggered the notification. The notification types for monitors specific to the type of monitor being used are:
- ***jmx.monitor.counter.threshold:*** Generated when a CounterMonitor's derived gauge meets or exceeds the configured threshold value
- ***jmx.monitor.gauge.high:*** Generated when a GaugeMonitor's derived gauge meets or exceeds the configured high threshold
- ***jmx.monitor.gauge.low:*** Generated when a GaugeMonitor's derived gauge decreases to or below the configured low threshold
- ***jmx.monitor.string.matches:*** Generated when a StringMonitor's derived gauge first matches the configured string to compare
- ***jmx.monitor.string.differs:*** Generated when a StringMonitor's derived gauge first differs from the configured string to compare

Notifications are also generated when certain error conditions are encountered. The common set of error types are:
- ***jmx.monitor.error.mbean:*** Generated when one of the observed MBeans is not registered with the MBean server
- ***jmx.monitor.error.attribute:*** Generated when the observed attribute does not exist in one of the observed MBeans
- ***jmx.monitor.error.type:*** Generated when the observed attribute value is null or not the appropriate type for the type of monitor being used
- ***jmx.monitor.error.runtime:*** Generated when other errors are encountered while trying to get the value of the observed attribute
- ***jmx.monitor.error.threshold:*** Generated by counter or gauge monitors when the configured threshold parameters are not of the same type as the observed attribute

To use a monitor, you need to create an instance of the appropriate Monitor MBean and register it with the MBean server, configure it, add one or more listeners, and start the monitor. Creating the Monitor MBean involves creating a name for the new MBean and invoking one of the createMBean() methods on the MBeanServer:

```
String monitorName = "mydomain:Name=MyMonitor";
ObjectName monitorMBeanName = new
ObjectName(monitorName);
ObjectInstance oiMonitor =
mbeanServer.createMBean("javax.management.moni-
tor.GaugeMonitor", monitorMBeanName);
```

Using the MBeanServer.createMBean() method also registers the MBean with the MBean server. Next, we need to configure the Monitor MBean using the appropriate attributes or operations. For our JMXMonitor example, we use a GaugeMonitor and configure its attributes using the MBeanServer.setAttributes() method (see Listing 3).

Next, we invoke the setThresholds() operation:

```
Object[] params = new Object[2];
params[0] = new Integer(10);
params[1] = new Integer(1);
String[] signature = new String[2];
signature[0] = "java.lang.Number";
signature[1] = "java.lang.Number";
Object retval =
    mbeanServer.invoke(monitorMBeanName,
"setThresholds", params, signature);
```

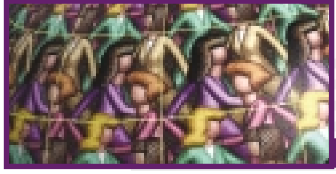
Then we add the listener to the monitor:

```
MyMonitorListener me = new MyMonitorListener();
mbeanServer.addNotificationListener(monitorMBean-
Name,
     me, null, null);
```

Finally, we start the monitor:

```
params = new Object[0];
signature = new String[0];
retval = mbeanServer.invoke(monitorMBeanName,
"start",
                             params, signa-
ture);
```

Now, whenever the PendingRequestCurrentCount attribute of the weblogic.kernel.Default execute queue first exceeds 10 or first falls below 1, a notification message



will be sent to the listener. Once an application is through with a monitor, it should stop the monitor and unregister it from the MBean server:

```
retval = mbeanServer.invoke(monitorMBeanName,
"stop",
     params, signature);
mbeanServer.unregisterMBean(monitorMBeanName);
```

If you fail to do this, the next time the application starts up it will fail unless it uses a different name for the Monitor MBean or the server has been restarted. For more information about JMX monitors, see the JMX 1.0 specification and Javadocs. JMX provides one final notification mechanism, known as the JMX Timer service, that provides the ability to be notified at a specific date and time or even at periodic intervals.

JMX Timers

The JMX timer service generates notifications in two different ways:

- Single occurrence notifications
- Repeated notifications that occur at regular intervals over a specified period of time and/or number of occurrences

Like the monitor MBean, the Timer MBean is created on the MBeanServer and configured to generate these timed notifications. Notifications generated by the Timer MBean are sent using an instance of the TimerNotification class. To create a notification, use one of the Timer MBean's addNotification() methods, which require some or all of the following arguments:

- ***type:*** String used to represent the type of notification
- ***message:*** String used to send detailed information about the notification
- ***userData:*** Optional handback object
- ***date:*** Date class used to specify when the notification should occur
- ***period:*** Interval in milliseconds between notifications (0 or null for nonrepeating notifications)
- ***nbOccurences:*** Total number of times that the notification will occur (0 or null means that it repeats indefinitely if the period is not 0 or null)

The code to create, configure, add a listener to, and start the Timer MBean looks very much like the code we used earlier when working with the JMX monitor. In the same way, your application should stop and unregister your Timer MBean when it is finished with it so that there will be no naming collisions caused by trying to create an MBean twice with the same name. Rather than list the code here, please see the downloadable examples.

Summary

In this article, we showed you how to use JMX notifications, monitors, and timers. JMX notification provides the ability for applications to register their interest in certain events and receive callbacks when those events occur. Using JMX notification with a NotificationListener provides a simple mechanism to detect predefined events generated by MBeans. Monitors provide a more sophisticated way of observing the value of an MBean attribute and receiving notification when the value of the attribute exceeds the configured threshold. JMX Timers provide a mechanism to generate a notification at a future date and time or to generate notifications at regular intervals. These JMX services provide a management application with the ability to monitor the behavior of a managed application and respond to changes without the need for human intervention.

The next and final article in this series will dive into the details of creating custom MBeans and extending the Admin Console to display them.

Listing 1

```
package wldj;

import java.io.Serializable;
import javax.management.AttributeChangeNotification;
import javax.management.Notification;
import javax.management.NotificationListener;

public class MyAttributeChangeListener
    implements javax.management.NotificationListener, Serializable
{
    String mbeanName;

    public MyAttributeChangeListener(String mbeanName)
    {
        this.mbeanName = mbeanName;
    }

    public void handleNotification(Notification n, Object handback)
    {
        if (n instanceof AttributeChangeNotification) {
            AttributeChangeNotification acn =
                (AttributeChangeNotification)notification;
            System.out.println("MBean " + mbeanName + " attribute " +
                               acn.getAttributeName() + " of type " +
                               acn.getAttributeType() +
                               " changed from " + acn.getOldValue() +
                               " to " + acn.getNewValue());
        }
        else {
            System.out.println("Unexpected notification type: " +
                               notification.getClass().getName());
        }
    }
}
```

Listing 2

```
package wldj;

import javax.management.Notification;
import javax.management.NotificationFilter;

public class MyAttributeChangeFilter implements NotificationFilter
{
    public boolean isNotificationEnabled(Notification n)
    {
        if (n instanceof AttributeChangeNotification)
            return true;
        else
            return false;
    }
}
```

Listing 3

```
AttributeList monitorAttributes = new AttributeList();
Attribute observedObjectAttribute =
    new Attribute("ObservedObject", queueMBeanName);
monitorAttributes.add(observedObjectAttribute);

Attribute observedAttributeAttribute =
    new Attribute("ObservedAttribute",
                  "PendingRequestCurrentCount");
monitorAttributes.add(observedAttributeAttribute);

...

monitorAttributes =
    mbeanServer.setAttributes(monitorMBeanName,
                              monitorAttributes);
```

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
BEA Systems	http://dev2dev.bea.com/medulla	800-817-4BEA	2-3
ColdFusion Developer's Journal	www.sys-con.com	888-303-5282	17
ColdFusion Resource CD	www.sys-con.com/freecd	888-303-5282	37
Compuware	www.compuware.com	800-COMPUWARE	11
Cyanea	www.cyanea.com/nomoremadness.wsa	877-CYANEA8	23-25
H&W Computer	www.hwcs.com/wldj1.asp	800-338-6692	9
Information Storage + Security Journal	www.issjournal.com	888-303-5282	31
IT Solutions Guide	www.sys-con.com	201-802-3021	35
Java Developer's Journal	www.sys-con.com	888-303-5282	27, 41
Mercury Interactive	www.mercuryinteractive.com/optimizej2EE	800-831-8911	13
MX Developer's Journal	www.sys-con.com/mx/subscription.cfm	888-303-5282	29
NetIQ	www.netiq.com/solutions/web	408-856-3000	52
Quest Software	http://java.quest.com/jcsv/wldj	800-663-4723	51
ReportingEngines	www.reportingengines.com	888-884-8665	21
SYS-CON Publications	www.sys-con.com	888-303-5282	39, 44
SYS-CON Publications	www.sys-con.com	888-303-5282	45, 49
Wily Technology	www.wilytech.com	888-GET-WILY	5

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

Considering MySQL? Read On... part 2
This month, we look at the changes, adjustments, and modifications needed to support Enterprise JavaBeans (EJBs), the core component model for J2EE, as well as the Java Transaction API (JTA). Just as in the April article, the development tool of choice is WebLogic Workshop (WLW) 8.1 and the application is deployed to WebLogic Server (WLS) 8.1.

Instrumenting a Java Page Flow Using JMX Technology
With Web services use rising, organizations are seeing a growing complexity in the enterprise systems being built. The need for a robust management solution is critical as organizations look for better ways to monitor and control their IT environment. This article demonstrates how JMX MBeans can be developed in BEA WebLogic Workshop, which has a simplified programming model, based on controls, events, and properties, that greatly enhances the development of J2EE and Web services.

The BEA WebLogic Messaging Bridge
A messaging system is one in which applications are loosely coupled through the exchange of messages - something like an e-mail system for applications.
A messaging bridge in turn moves messages between any two messaging systems/products. BEA WebLogic introduced a messaging bridge in WebLogic 6.1, and has enhanced the features with the later releases of WebLogic. We'll look at what it is, and how to use it.

Transactions: Driving You to Distraction
This month, Peter Holditch looks at a piece of technology that exists somewhere in the twilight zone between commodity and valuable asset - the humble JDBC driver. One of the main drivers for the widespread adoption of the J2EE platform was the existence of standards that provide a good degree of plug-and-playability in terms of the resources accessing and accessed by the platform. One principal such resource is the SQL database (whose adoption in its own life cycle was fueled by the standard relational model for data access, providing some swapability at a different layer of the architecture) the Java standard that makes SQL databases pluggable at this level is, of course, JDBC.

BEA WebLogic DEVELOPER'S JOURNAL

Transactions, Suspension, and the Ticking Clock

KEEP IT SHORT – AND DON'T TIME OUT

BY **PETER HOLDITCH**

This month's article is again inspired by a posting on the weblogic.developer.interest.transaction newsgroup. The question (excerpted from the posting) was:

Does the <trans-timeout-seconds>10</trans-timeout-seconds> in weblogic-ejb-jar.xml apply to transactions that are in a suspended state?
I have EJB1 (Container Managed/Required) that starts transaction T1 and does some work, then calls another EJB, EJB2 (Container Managed/NotSupported), which makes an interdomain T3 call. Since EJB2 is configured with NotSupported transaction attribute, transaction T1 is suspended for the time being while the business method of EJB2 is executing. The question is, if I set <trans-timeout-seconds>10</trans-timeout-seconds> for transactions started by EJB1, will it work?

The answer is yes. End of article.

Only joking! The subject of transaction timeouts and their relationship to the flow of control through an application's logic is one that often comes up, and is frequently misunderstood, so here goes another journey through "how the bits fit together."

First, a step back. As you will know if you have read my previous articles (or any other sources of information about distributed transactions), long-running XA transactions are a bad thing. Database (and potentially other) locks are held from the moment that a transaction touches the data until the point that it commits or aborts. Lots of locked data can quickly become a performance problem for any application, resulting ultimately in your whole database being locked. This is bad. The moral of that story is, keep your XA transactions as short as you possibly can to allow locks to be recycled as fast as possible.

In this context, the transaction timeout is your friend. You can set a timeout in a deployment descriptor, programmatically in your code, or via the JTA page in the BEA WebLogic console and the transaction manager will respect this timeout on your behalf. If a transaction lasts longer than the timeout permits, WebLogic's transaction manager will automatically go about rolling it back for you. The assumption is that some kind of application error (anything from a logic error to the hardware being temporarily overloaded, causing the application to run slowly) is causing the timeout to be exceeded, and that cleaning up the transaction (and thus freeing the associated locks) will help the overall health and well-being of the system.

So that's fine. That's what the transaction timeout is, and that's what it's for. Generally, tuning transaction timeout values such that transactions in a smoothly running system will never time out (by a small margin of safety) offers you a degree of protection from spikes in load and errors in logic that can turn little localized dramas into big, scary catastrophes.

As Usual, the Transaction Manager Is Your Friend

So far so clear. The confusion tends to lie with how the transaction timeout relates to other behaviors of the application server. The behaviors in question are how the timeout mechanism behaves with respect to a suspended transaction (the basis of the posting I opened with) and what happens to threads executing code on behalf of transactions that time out?

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

So what is suspension of a transaction? Recall that transactions have two halves. The synchronous runtime part keeps track of what application actions should be logically grouped into a transaction (controlled, by calls to begin, commit, etc.) and the asynchronous part whose obvious function is to take the list of resources collected by the runtime part and process the two-phase commit asynchronously with respect to the application flow. What the calls to UserTransaction.begin do is associate a transaction object with the current thread of execution. The fact that a transaction is associated with a thread of execution tells the application-server infrastructure to propagate the transaction along with any calls or data accesses the thread makes. A call to UserTransaction.commit disassociates the transaction object and the current thread and kicks off the two-phase commit of the (now-completed) transaction. What suspending a transaction does is simply break the association between a thread of execution and a transaction. Resuming it rebuilds the association. Thus, if you need to work outside of an active transaction, you suspend it, do the work, and then resume it when you're done.

There is one more necessary detail here: the life cycles of the synchronous and asynchronous components do not follow each other serially. Both begin concurrently when the UserTransaction.begin call is made. The synchronous part clearly ends before the asynchronous part, since the two-phase commit processing that is the latter's primary responsibility cannot begin until the synchronous piece has completed – completion signalled by calling commit.

Given the rationale I have laid out for the timeout's existence, it should be clear that the other duty performed by the asynchronous component of the transaction is timeout processing. From the moment the begin call is made, the timer starts ticking independently of the flow of your processing. When the timeout expires, the Transaction Manager will roll the transaction back. The first you hear of that happening may be when calls you make in the thread associated with the transaction start to throw exceptions, since it is not possible to propagate a rolled-back transaction (what would be the point, the results of the call will not persist under any circumstances?).

After all that, the direct answer to the question we started with is that suspending a transaction has no impact whatsoever on its timeout; it simply breaks the association between transaction and thread. If your code suspends a transaction and then goes to sleep for a week, it should expect to find that the transaction has timed out while it slept.

The Transaction Has Timed Out While It Slept

Finally, to the second relationship I mentioned – that between the timed-out transaction and the thread currently executing work on the transaction's behalf. Some people hope that if they set a transaction timeout on an operation, when the timeout is exceeded all work on behalf of the transaction will instantly stop. In an ideal world, that would happen (and WebLogic does all it can to try and stop you wasting cycles by refusing to propagate a timed-out transaction), but the world (and the world I am referring to here is the general world of java) is *not* ideal. So the transaction times out, and WebLogic wants to stop your thread. What can it do? Looking at the java.lang.Thread class, we quickly conclude that it can't do a whole lot. It could call Thread.interrupt, but then code inside the thread would have to catch the Interrupt and clean up – so this requires cooperation for the application code. It could call Thread.destroy, but according to the Javadoc this is not implemented (and if it was, would leave no way to clean up resources). Or it could call Thread.stop() except that that's deprecated with a mile-long health warning that talks about how stopping a thread with no controlled cleanup could lead to all sorts of inconsistent data being left around.

There is no easy answer – there isn't really a facility in J2SE or J2EE as they stand today to allow a thread to be safely and asynchronously terminated. WebLogic does what it can – the node manager will try and detect "stuck" threads to warn operators about them; the runtime will try and prevent work continuing on behalf of rolled-back transactions by throwing exceptions rather than propagating them. And, it does all it can at the JDBC level (such as calling Statement.cancel for statements running on behalf of rolling back transactions where possible) but at the end of the day – until the thread returns control to the container – there isn't much more WebLogic can do than wait.

In summary, the transaction timeout impacts the asynchronous behavior of the transaction manager only – it has no direct impact whatsoever on the flow of control within the application logic.

I have timed out again for another month. Roll back and read more next time!

A LIMITED TIME SAVINGS

SUBSCRIBE TODAY TO MULTI

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

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

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

Pick the CDs to go with your Multi-Pack order

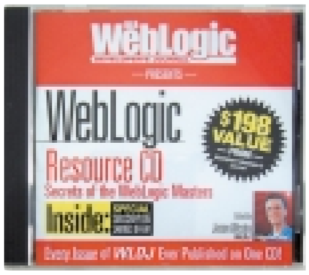
- Pick one CD with your 3-Pack order
- Pick two CDs with your 6-Pack order
- Pick three CDs with your 9-Pack order

☐ Web Services Resource CD
☐ Java Resource CD
☐ WebLogic Resource CD
☐ ColdFusion Resource CD
☐ XML Resource CD
☐ WebSphere Resource CD
☐ Linux Resource CD

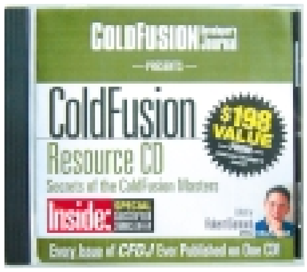
Your order will be processed the same day!

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

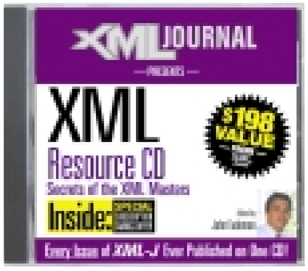
The most complete library of exclusive *JDJ* articles compiled on one CD! Assembled by *JDJ* Editor-in-Chief Alan Williamson, more than 1,400 exclusive articles are organized into over 50 chapters, including fundamentals, applets, advanced Java topics, Swing, security, wireless Java,... and much more!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



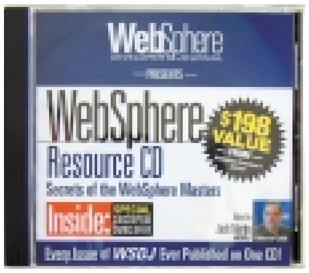
The most complete library of exclusive *WLDJ* articles ever assembled! More than 200 articles provide invaluable information on "everything WebLogic", including WebLogic Server, WebLogic Portal, WebLogic Platform, WebLogic Workshop, Web services, security, migration, integration, performance, training...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


COLDFUSION Developer's Journal
ColdFusion Resource CD
$198 VALUE
Every Issue of CFDJ Ever Published on One CD!

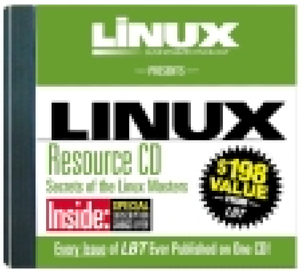
The most complete library of exclusive *CFDJ* articles on one CD! This CD, edited by *CFDJ* Editor-in-Chief Robert Diamond, is organized into more than 30 chapters with more than 400 exclusive articles on CF applications, custom tags, database, e-commerce, Spectra, enterprise CF, error handling, WDDX...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



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



The most up-to-date collection of exclusive *WSDJ* articles! More than 200 articles offer insights into all areas of WebSphere, including Portal, components, integration, tools, hardware, management, sites, wireless programming, best practices, migration...
LIST PRICE $198



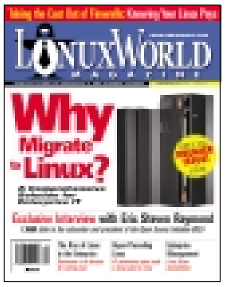
An up-to-the-minute collection of exclusive *Linux Business & Technology* articles plus Maureen O'Gara's weekly *LinuxGram*, which keeps a finger on the pulse of the Linux business community. Edited by *LBT's* Editor-in-Chief Alan Williamson, these articles cover migration, hardware, certification, and the latest Linux-related products. Available June 2003!
LIST PRICE $198

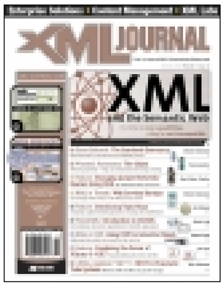
Subscribe Online Today

IPLE MAGAZINES ONLINE

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS





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 World Magazine**

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

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

www.sys-con.com/2001/sub.cfm

SYS-CON MEDIA

XA Transactions

NEEDED MORE OFTEN THAN YOU THINK

Most developers have at least heard of XA, which describes the standard protocol that allows coordination, commitment, and recovery between transaction managers and resource managers.

BY **WES HEWATT**

AUTHOR BIO…

Wes Hewatt has over 14 years of experience designing and deploying mission-critical applications for Fortune 1000 companies. As a senior systems engineer for BEA Systems, Wes works with BEA's customers to develop J2EE applications for the BEA WebLogic Platform. He specializes in Web services and integration technologies.

CONTACT...

whewatt@bea.com

Products such as CICS, Tuxedo, and even BEA WebLogic Server act as transaction managers, coordinating transactions across different resource managers. Typical XA resources are databases, messaging queuing products such as JMS or WebSphere MQ, mainframe applications, ERP packages, or anything else that can be coordinated with the transaction manager. XA is used to coordinate what is commonly called a two-phase commit (2PC) transaction. The classic example of a 2PC transaction is when two different databases need to be updated atomically. Most people think of something like a bank that has one database for savings accounts and a different one for checking accounts. If a customer wants to transfer money between his checking and savings accounts, both databases have to participate in the transaction or the bank risks losing track of some money.

The problem is that most developers think, "Well, my application uses only one database, so I don't need to use XA on that database." This may not be true. The question that should be asked is, "Does the application require shared access to multiple resources that need to ensure the integrity of the transaction being performed?" For instance, does the application use Java 2 Connector Architecture adapters, the BEA WebLogic Server Messaging Bridge, or the Java Message Service (JMS)? If the application needs to update the database and any of these other resources in the same transaction, then both the database and the other resource need to be treated as XA resources.

In addition to Web or EJB applications that may touch different resources, XA is often needed when building Web services or BEA WebLogic Integration applications. Integration applications often span disparate resources and involve asynchronous interfaces. As a result, they frequently require 2PC. An extremely common use case for WebLogic Integration that calls for XA is to pull a message from WebSphere MQ, do some business processing with the message, make updates to a database, and then place another message back on MQ. Usually this whole process has to occur in a guaranteed and transactional manner. There is a tendency to shy away from XA because of the performance penalty it imposes. Still, if transaction coordination across multiple resources is needed, there is no way to avoid XA. If the requirements for an application include phrases such as "persistent messaging with guaranteed once and only once message delivery," then XA is probably needed.

Figure 1 shows a common, though extremely simplified, BEA WebLogic Integration process definition that needs to use XA. A JMS message is received to start the process. Assume the message is a customer order. The order then has to be placed in the order shipment database and placed on another message queue for further processing by a legacy billing application. Unless XA is used to coordinate the transaction between the database and JMS, we risk updating the shipment database without updating the billing application. This could result in the order being shipped, but the customer might never be billed.

Once you've determined that your application does in fact need to use XA, how do we make sure it is used correctly? Fortunately, J2EE and the Java Transaction API (JTA) hide the implementation details of XA. Coding changes are not required to enable XA for your application. Using XA properly is a matter of configuring the resources that need to be enrolled in the same transaction. Depending on the application, the BEA WebLogic Server resources that most often need to be configured for XA are connection pools, data sources, JMS Servers, JMS connection factories, and messaging bridges. Fortunately, the entire configuration needed on the WebLogic side can be done from the WebLogic Server Console.

Before worrying about the WebLogic configuration for XA, we have to ensure that the resources we want to access are XA enabled. Check with the database administrator, the WebSphere MQ administrator, or whoever is in charge of the resources that are outside WebLogic. These resources do not always enable XA by default, nor do all resources support the X/Open XA interface, which is required to truly do XA transactions. For example, some databases require that additional scripts be run in order to enable XA.

For those resources that do not support XA at all, some transaction managers allow for a "one-phase" optimization. In a one-phase optimization, the transaction manager issues a "prepare to commit" command to all of the XA resources. If all of the XA resources respond affirmatively, the transaction manager will commit the non-XA resource. The transaction manager will then commit all of the XA resources. This allows the transaction manager to work with a non-XA resource, but normally only one XA resource per transaction is allowed. There is a small chance that something will go wrong after committing the non-XA resource and before the XA resources all commit, but this is the best alternative if a resource just doesn't support XA.

Connection pools are where most people start configuring WebLogic for XA. The connection pool needs to use an XA driver. Most database vendors provide XA drivers for their databases. BEA WebLogic Server 8.1 SP2 ships with a number of XA drivers for Oracle, DB2, Informix, SQL Server, and Sybase. We need to ensure that the Driver classname on the connection pool page of the BEA WebLogic Console is in fact an XA driver. When using the configuration wizards in BEA WebLogic Server 8.1, the wizards always note which drivers are XA enabled.

When more than one XA driver is available for the database involved, be sure to run some benchmarks to determine which driver gives the best performance. Sometimes different drivers for the same database implement XA in completely different ways. This leads to wide variances in performance. For example, the Oracle 9.2 OCI Driver implements XA natively, while the Oracle 9.2 Thin Driver relies on stored procedures in the database to implement XA. As a result, the Oracle 9.2 OCI driver generally performs XA transactions much faster than the Thin driver. Oracle's newest Type 4 driver, the 10*g* Thin Driver, also implements XA natively and is backwards compatible with some previous versions of the Oracle database. Taking the time to fully evaluate alternative drivers can lead to significant performance improvements.

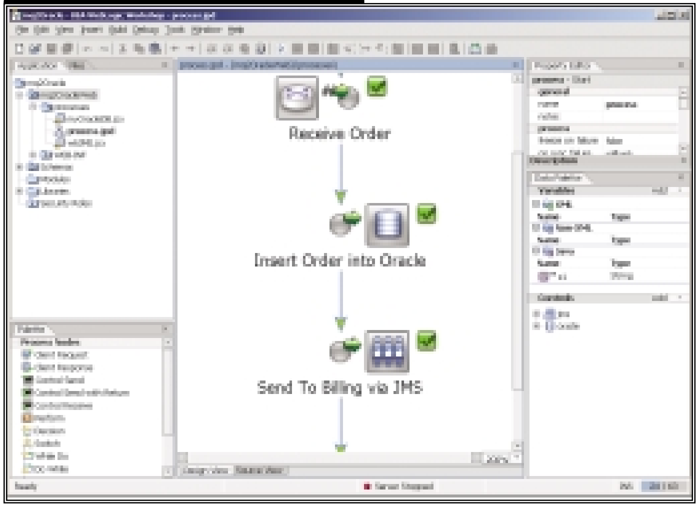
If only some of the database access needs to be done under XA, create two connection pools for the same database. Use an XA driver on one of the connection pools and a non-XA driver on the other. This will avoid the performance overhead of XA transactions for database calls that don't need 2PC.

Closely related to the connection pools are the data sources. In order to use XA, a data source must have the value "Honor Global Transactions" set to true. Prior to BEA WebLogic Server 8.1, these data sources appeared in the Console under the heading "Tx Data Sources". In 8.1, all data sources are under the same heading. Turning this flag on means that BEA WebLogic Server will be able to correctly handle transactions in a number of different scenarios. Setting this flag will ensure that WebLogic Server's JTA implementation will automatically enroll the data source in an XA transaction if it is required. There are also situations where this flag should be set even if your application does not use XA. The "Honor Global Transactions" flag should also be enabled if your application makes any explicit JTA calls, uses container-managed transactions with EJBs, or issues multiple SQL statements within the same transaction. In these non-XA situations, BEA WebLogic Server will ensure that the application retains the proper database connection from the connection pool to ensure transactional integrity.

A second flag on the data source page that is occasionally used is the "Emulate Two-Phase Commit for Non-XA Driver" setting. This flag should only be used if an XA driver cannot be obtained for the database. When this flag is on, a one-phase optimization is used. BEA WebLogic Server will first issue the "prepare to commit" command to the XA resources, commit the database that has emulation enabled, and then commit the resources in the transaction that are XA enabled. As long as nothing goes wrong, the data will still be consistent.

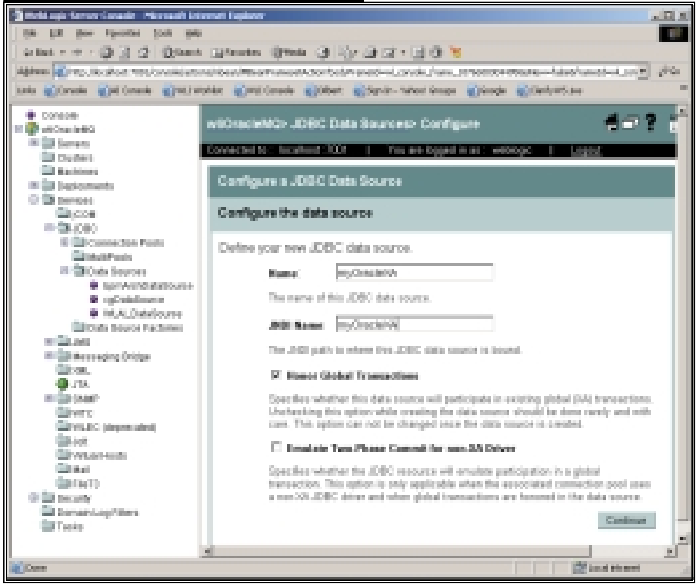
There is a potential that WebLogic Server will commit the non-XA transaction, only to have the transaction on the XA resource

FIGURE 1



A BEA WebLogic Integration process that requires global resources

FIGURE 2


wlOracleMQ> JDBC Data Sources> Configure

Configure a JDBC Data Source

Configure the data source

Define your new JDBC data source.

Name: myOracleXA

The name of this JDBC data source.

JNDI Name: myOracleXA

The JNDI path to where this JDBC data source is bound.

[✓] Honor Global Transactions

Specifies whether this data source will participate in existing global (XA) transactions. Unchecking this option while creating the data source should be done rarely and with care. This option can not be changed once the data source is created.

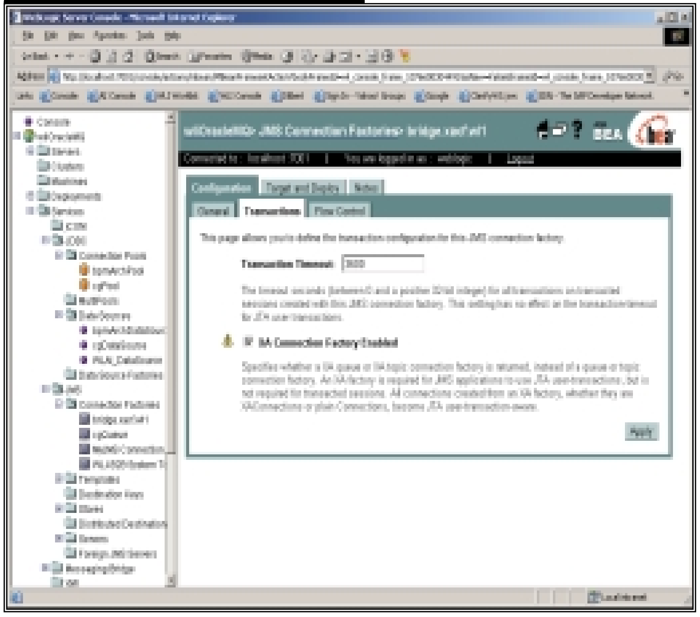
[ ] Emulate Two-Phase Commit for non-XA Driver

Specifies whether the JDBC resource will emulate participation in a global transaction. This option is only applicable when the associated connection pool uses a non-XA JDBC driver and when global transactions are honored in the data source.

Continue

New data source configured to support XA transactions

FIGURE 3


wlOracleWID> JMS Connection Factories> bridge.xacf.wf1

Connected to : localhost 7001 | You are logged in as : weblogic | Logout

This page allows you to define the transaction configuration for this JMS connection factory.

Transaction Timeout: 3600

The timeout seconds (between 0 and a positive 32-bit integer) for all transactions on transacted sessions created with this JMS connection factory. This setting has no effect on the transaction timeout for JTA user transactions.

☑ XA Connection Factory Enabled

Specifies whether a XA queue or XA topic connection factory is returned, instead of a queue or topic connection factory. An XA factory is required for JMS applications to use JTA user-transactions, but is not required for transacted sessions. All connections created from an XA factory, whether they are XAConnections or plain Connections, become JTA user-transaction-aware.

A JMS connection factory with XA enabled

fail. WebLogic Server allows only one data source using emulation per transaction. Given the availability of XA drivers for most databases and the potential for inconsistent data, this setting should rarely be used. Figure 2 shows a data source properly configured for XA.

Within BEA WebLogic Server, JMS Servers themselves are XA resources. There is nothing special that needs to be configured to XA enable a JMS Server, but there is one configuration item that seems counter-intuitive. When using a JDBC store for the JMS Server, you might think that the connection pool used by the JDBC store needs to use an XA driver. In fact, the exact opposite is true. The connection pool for the JDBC store should *not* use an XA driver. In this case, the XA resource is the JMS Server, not the database. For this reason, a JMS Server that uses a file store is still capable of participating in an XA transaction. The decision about whether to use a file store or a JDBC store for a JMS Server should not be based on whether or not an application will need to use XA.

The next step to ensure that you are using XA with your JMS-based application is to use an XA connection factory. Again, the application code does not change, but a configuration setting in the BEA WebLogic console needs to be checked. After creating a new connection factory, you need to go to the "Transactions" tab and check "XA Connection Factory Enabled". Changing this value will require a server restart. If only some of the application's work with JMS needs to use XA, you may want to create another connection factory that does not use XA. Figure 3 shows a JMS connection factory configured for XA transactions.

The last resource that deserves mention is the messaging bridge, which was introduced originally for BEA WebLogic Server 7.0 and is intended to make it easier to integrate WebLogic JMS with foreign JMS providers. The messaging bridge acts as a bidirectional store-and-forward mechanism to transfer messages back and forth between WebLogic JMS and another messaging product, such as WebSphere MQ. WebLogic applications do not interact directly with the messaging bridge. Instead, they interact in a normal manner with the local WebLogic JMS queues or topics. The local queue or topic is then bridged to the foreign JMS provider. However, there are several configuration settings for the messaging bridge that need to be set correctly to ensure "guaranteed once and only once" delivery between WebLogic and the other product. Some of the settings are on the JMS Bridge Destination and some of the settings are on the bridge itself. These settings affect whether or not BEA WebLogic Server will use XA transactions when it transfers messages.

When configuring a Bridge Destination, there are three settings that control whether or not WebLogic Server will treat the destination as an XA resource: the adapter, the Adapter JNDI Name, and the Connection Factory JNDI Name. BEA WebLogic Server uses J2EE Connector Architecture adapters to communicate with the Bridge Destinations. The adapter that is used must support XA. BEA WebLogic Server provides a generic XA adapter named jms-xa-adp.rar. The Adapter JNDI Name for this is eis.jms.WLSConnectionFactoryJNDIXA. Finally, the foreign JMS Server must have an XA-enabled connection factory, and the name of this connection factory is placed in the Connection Factory JNDI Name field.

The messaging bridge itself has different qualities of service available: Exactly-Once, Atmost-once, and Duplicate-okay. If "guaranteed once and only once" delivery is a requirement for the application, then the only acceptable setting for Quality of Service on the messaging bridge configuration page is "Exactly-Once". The QoS Degradation Allowed flag should also be unchecked. Checking this box allows BEA WebLogic Server to default to a lower quality of service if it is unable to get an XA connection to the foreign provider. This is usually a very bad idea. Qualities of service should be dictated by the business requirements. Business requirements are rarely flexible enough to switch back and forth between "Exactly-Once" and other service levels. Using the QOS Degradation Allowed flag means that no one can predict which quality of service WebLogic Server will be using at runtime.

Once you've configured XA for the resources involved in an application, how can you determine that everything is working properly? Under normal conditions, where all resources are available and operating correctly, a non-XA–enabled application will behave exactly the same as an XA-enabled one. XA proves its value when an application encounters unexpected situations. The test plan for an application should include scenarios where each resource is unavailable. Testing should also evaluate what happens when a resource becomes unavailable in the middle of processing transactions. Intentionally killing a BEA WebLogic Server instance, causing a duplicate key error, or restarting a database simulates situations that can happen in production. If XA has been properly configured, all resources should complete or roll back the same transactions.

Conclusion

By now two things should be clear. XA transactions are needed more often than most developers realize, and XA is very easy to configure within BEA WebLogic Server. Always evaluate the configuration based on the application's business requirements and then choose the appropriate settings to make sure that transactions behave in the way they should.

SYS-CON MEDIA

304,187 of the World's Foremost IT Professionals

DIRECT MAIL, EMAIL OR MULTI-CHANNEL

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

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

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

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

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

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

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

SPECIAL INSERT
COLDFUSION Developer's Journal
Windows Server 2003 and CFMX 6.1

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

MOVING AHEAD FOR THE 21ST CENTURY
BEA WebLogic DEVELOPER'S JOURNAL

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

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

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

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

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

e-POSTDIRECT® Your iMarketing Source

For email information:
contact Frank at 845-731-3832
frank.cipolla@epostdirect.com
epostdirect.com 800-409-4443 fax845-620-9035

For postal information:
contact Kevin at 845-731-2684
kevin.collopy@edithroman.com
edithroman.com 800-223-2194 fax845-620-9035

List Brokerage & Management
EDITH ROMAN
Data Processing • Email Marketing

News & Developments

H&W's Performance Scouts Offer Insight into Mainframe Resources

(Boise, Idaho) – H&W has announced the release of DiagnoSys Performance Scouts, comprehensive mainframe data collection agents for its DiagnoSys intelligent performance management solution. Together, DiagnoSys and Performance Scouts ensure that crucial Web applications perform optimally, keeping users satisfied and revenue flowing.

Unlike competing products, DiagnoSys is an intelligent performance management solution that examines application performance from end to end. The systematic use of DiagnoSys in preproduction and production environments supplies automated analysis to identify and correct the most complex performance problems. DiagnoSys offers a sophisticated combination of data collection, correlation, and analysis.
www.hwcs.com

Interwoven Maximizes Sales Profitability Through Streamlined Channel Management Offering

(Sunnyvale, CA) – Interwoven Inc., a next-generation enterprise content management company, is driving sales productivity and increased revenue through the Interwoven Channel Management (ICM) solution. This solution connects businesses with their direct and indirect channels through information sharing, business processes, and sales applications to streamline the sales process and realize more profitable channels. The ICM solution integrates with market-leading and custom developed Web portals, mobile devices, and e-mail systems to expand the profit potential of agents, account executives, resellers, and distributors while reducing the cost to support them.

The ICM solution integrates with leading portal providers such as BEA, SAP, and custom-developed systems. ICM leverages Interwoven's flagship TeamSite Content Server, Interwoven MetaTagger, and Interwoven OpenDeploy to deliver a complete channel management content solution.

www.interwoven.com

Hijos de Rivera S.A. Selects BEA WebLogic Platform 8.1

(San Jose, CA) – Hijos de Rivera S.A., one of Spain's leading breweries and owner of the well-known brand Estrella de Galicia, has selected BEA WebLogic Platform 8.1 as the foundation for the company's technological infrastructure. Hijos de Rivera's goals are to improve and automate business processes and integrate new applications, with plans to enable the development of the company's consolidation and international expansion strategy.

Hijos de Rivera needed to overcome technologies that limited accessibility to data and an inability to integrate data into its information systems, many of which were formerly under the ownership and responsibility of each department. One prerequisite for the new solution was that it should utilize information already stored to help enable a smooth transition to the new environment. Hijos de Rivera selected BEA WebLogic Platform 8.1 as its integral J2EE solution, one designed to respond to the criteria regarding the independence of data sources and compliance with standards, and a high reuse of components.
www.bea.com

BEA, Documentum, MobileAware Announce Mobile Content Delivery Solution

(Cannes, France) – BEA Systems; Documentum, a division of EMC; and MobileAware have announced an integrated product set for mobile content delivery that can enable consumers to use their mobile phones to order and purchase online content and downloadable multimedia. In addition, the content delivery suite is designed to manage the acquisition of internal and third-party content and the provisioning of that content to different customer market segments.

The integrated products are a combination of MobileAware's Mobile Interaction Server version 3.2, BEA WebLogic Platform 8.1 and the Documentum 5 Enterprise Content Management platform, all running on Intel-based servers. The combined product set can enable mobile operators, content aggregators, and system integrators to rapidly deploy the product set and customize the functionality to meet their requirements.
www.bea.com,
www.documentum.com,
www.mobileaware.com

BEA WebLogic 8.1 Momentum Grows

(San Jose, CA) – BEA Systems, Inc., has announced that total revenues exceeded $1 billion in its fiscal year ended Jan. 31, 2004, with adoption of BEA WebLogic Platform 8.1 continuing to gain traction as companies seek a scalable, adaptable platform for service-oriented architectures (SOA).

Customers are accelerating time-to-value from IT investments, reducing labor costs, extending the usable life of legacy IT assets, and bringing new Web applications to market rapidly by deploying SOAs on BEA WebLogic Platform 8.1. BEA saw particularly broad adoption in the quarter for its application infrastructure software in the energy, financial services, manufacturing, retail, telecommunications, government, and transportation sectors. www.bea.com

VERITAS Raises Bar on J2EE Application Performance Management

(Mountain View, CA) – VERITAS Software Corporation, a leading storage software provider, has announced that J2EE-based applications can now be migrated from development into production faster with the new adaptive instrumentation capabilities of VERITAS Indepth for J2EE software.

VERITAS Indepth for J2EE – Adaptive Instrumentation software is expected to dramatically reduce the time it takes to isolate J2EE application performance problems from weeks to minutes, and is the first software to automatically provide an in-depth way to isolate J2EE performance bottlenecks without adding system "overhead" that itself becomes another performance issue.
www.veritas.com

Generate dynamic rich content in your server-side Java applications

JClass® ServerViews

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

JClass ServerChart

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

JClass ServerReport

Generate high-quality Adobe PDF for a global audience.

QUEST SOFTWARE®

Evaluate and experience JClass today – visit:

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

Post-launch is NOT the time to be verifying web applications.

The wild blue yonder of operational monitoring and management is extremely unforgiving. Which means that going live with the monitoring software you used in development is a great way to go dead—quickly! You simply can't support operations if your staff is drowning in details provided by development profiling tools and debuggers. **Let NetIQ cover your apps...with AppManager.**

AppManager—the industry's easiest-to-use Systems Management suite—is a proven management system for monitoring J2EE application servers, databases, operating systems and even end-user response time. NetIQ's AppManager monitors ALL application components—not just your server. **NetIQ. Nobody does UNIX better. Nobody.**

Visit us at www.netiq.com/solutions/web to learn how we can help you address the challenges of your operational monitoring and management.

netiQ®
Work Smarter®